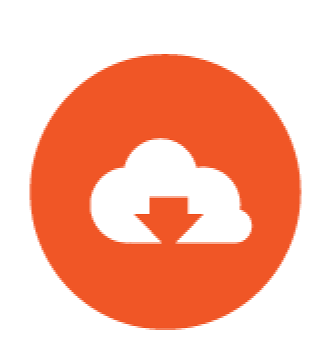
PHP+MySQL动态网站开发从入门到精通

（视频教学版）

张工厂　编著



- PHP 7+MySQL 5.7最佳组合
- 提供大量开发示例，让你看得懂、学得会、做得出
- 将最实用的技巧融入到每个案例中，教你快速成为网站应用开发高手



源代码、课件、教学视频、命令速查手册

清华大学出版社

PHP+MySQL 动态网站开发从入门到精通（视频教学版）

张工厂　编著



清華大学出版社
北京

内 容 简 介

本书循序渐进地介绍了 PHP 7 开发动态网站的主要知识和技能，提供了大量的 PHP 和 MySQL 的应用实例供读者实践。每一章都清晰地讲述了代码作用及其编写思路，使读者能在最短时间内迅速掌握 PHP 的应用开发技能。

全书共 21 章，分别介绍了 PHP 7 的基本概念、PHP 服务器环境配置、PHP 的基本语法、PHP 的语言结构、字符串和正则表达式、数组、时间和日期、面向对象、错误处理和异常处理、PHP 与 Web 页面交互、文件与目录操作、图形图像处理、Cookie 与会话管理、MySQL 数据库基础、数据表的基本操作、数据的基本操作、数据库的备份与还原、PHP 操作 MySQL 数据库等，最后通过两个综合案例，使读者进一步巩固所学的知识，提高综合实战能力。

本书适合 PHP 和 MySQL 的初学者，以及广大网站开发人员，也可供高等院校和培训学校相关专业的师生参考。

图书在版编目（CIP）数据

PHP+MySQL动态网站开发从入门到精通：视频教学版/张工厂编著. — 北京：清华大学出版社，2017

ISBN 978-7-302-45451-9

I. ①P… II. ①张… III. ①PHP语言－程序设计②关系数据库系统 IV. ①TP312②TP311.138

中国版本图书馆CIP数据核字（2016）第274717号

责任编辑：夏毓彦
封面设计：王　翔
责任校对：闫秀华
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　张：**26.5　　**字　数：**678 千字
版　　次：2017 年 1 月第 1 版　　**印　次：**2017 年 1 月第 1 次印刷
印　　数：1~3500
定　　价：69.00 元

产品编号：070171-01

前　言

PHP+MySQL 的组合是目前世界上最为流行的 Web 开发工具。由于大型互联网站广泛使用这种开发技术，目前学习和关注 PHP+MySQL 的人越来越多，本书作者针对 PHP+MySQL 的初学者，从通俗易懂、容易入门和案例实用的目标出发，组织有丰富经验的开发人员，编写了这本 PHP+MySQL 动态网站开发的教材。

本书内容

全书共 21 章，主要介绍了 PHP 7 的基本概念、PHP 服务器环境配置、PHP 的基本语法、PHP 的语言结构、字符串和正则表达式、数组、时间和日期、面向对象、错误处理和异常处理、PHP 与 Web 页面交互、文件与目录操作、图形图像处理、Cookie 与会话管理、MySQL 数据库基础、数据表的基本操作、数据的基本操作、数据库的备份与还原、PHP 操作 MySQL 数据库等，最后通过两个综合案例，使读者进一步巩固所学的知识，提高综合实战能力。

本书特色

- 知识全面：涵盖了所有 PHP+MySQL 开发的知识点，读者可以由浅入深地掌握 PHP+MySQL 动态网站开发技术。
- 图文并茂：注重操作，在介绍案例的过程中，每一个操作均有对应的插图。这种图文结合的方式使读者在学习过程中能够直观、清晰地看到操作的过程以及效果，便于更快地理解和掌握。
- 易学易用：颠覆传统“看”书的观念，变成一本能“操作”的图书。
- 案例丰富：把知识点融汇于系统的案例实训当中，并且结合经典案例进行讲解和拓展。进而达到“知其然，并知其所以然”的效果。
- 提示技巧：本书对读者在学习过程中可能会遇到的疑难问题以“提示”和“技巧”的形式进行了说明，以免读者在学习的过程中走弯路。
- 技术实用：本书所有案例都是模仿现实网站开发而设计，通过本书最后两个综合案例，让读者快速创建动态的 PHP+MySQL 企业网站。

读者对象

本书是一本完整介绍 PHP+MySQL 动态网站开发技术的教程，内容丰富，条理清晰，实用性强，适合如下读者学习使用：

- 对 PHP+MySQL 动态网站制作有兴趣的初学者，可以快速入门。
- 对 PHP 语言感兴趣的初学者，可以快速掌握 PHP 语言开发基本技巧。
- MySQL 数据库的初学者，可以快速掌握 MySQL 的基本操作方法。
- PHP+MySQL 架构的 Web 系统开发人员。

致谢

本书主要有张工厂主编，参与本书编写人员还有胡同夫、王英英、刘增杰、肖品、孙若淞、王攀登、王维维、梁云亮、刘海松、陈伟光、刘增产、包惠利和刘玉萍等。本书虽然倾注了编者的努力，但由于水平有限、时间仓促，书中难免有疏漏之处，请读者谅解，如果遇到问题或有意见和建议，敬请与我们联系，我们将全力提供帮助，技术支持 QQ 群：2606372761。

源码、课件和教学视频下载

本书配套源码、课件和教学视频下载地址（注意数字和字母大小写）如下：

http://pan.baidu.com/s/1hswmnYG

如果下载有问题，请电子邮件联系 booksaga@163.com，邮件主题为“PHP+MySQL 编程”。

编者

2016 年 11 月

目　录

第 1 章　初识 PHP

在学习 PHP 之前，读者需要了解 PHP 的基本概念、PHP 的特点、PHP 开发常用工具等知识，通过本章的学习，读者可对 PHP 有一个初步的了解。本章将主要讲述入门 PHP 的基本知识。

本章学习目标

- 了解 PHP 的来龙去脉
- 熟悉 PHP 的应用领域
- 熟悉 PHP 7 新特点
- 掌握 PHP 常用开发工具

1.1　PHP 的发展

PHP 语言和其他语言有什么不同呢？对于此问题，首先需要了解 PHP 的概念和发展历程。

1.1.1　PHP 的概念

PHP 全名为 Personal Home Page，是英文 Hypertext Preprocessor（超级文本预处理语言）的缩写。P 是一种 HTML 内嵌式的语言，在服务器端执行的嵌入 HTML 文档的脚本语言，语言风格类似于 C 语言，被广泛用于动态网站的制作中。PHP 语言借鉴了 C 和 Java 等语言的部分语法，并有自己独特的特性，使 Web 开发者能够快速地编写动态生成页面的脚本。对于初学者而言，PHP 的优势是可以快速入门。

与其他的编程语言相比，PHP 是将程序嵌入到 HTML 文档中去执行，执行效率比完全生成 HTML 标记的方式要高许多。PHP 还可以执行编译后的代码，编译可以起到加密和优化代码运行的作用，使代码运行得更快。另外，PHP 具有非常强大的功能，所有的 CGI 功能 PHP 都能实现，而且几乎支持所有流行的数据库和操作系统。最重要的是，PHP 还可以用 C、C++进行程序的扩展。

1.1.2　PHP 的发展历程

目前，市面上有很多 Web 开发语言，其中 PHP 是比较出众的一种 Web 开发语言。与其他脚本语言不同，PHP 是通过全世界免费代码开发者共同的努力，才发展到今天的规模。要想了解 PHP，首先从它的发展历程开始。

在 1994 年，Rasmus Lerdorf 首次设计出了 PHP 程序设计语言。1995 年 6 月，Rasmus Lerdorf 在 Usenet 新闻组 comp.infosystems.www.authoring.cgi 上发布了 PHP 1.0 声明。这个早期版本提供了访客留言本、访客计数器等简单的功能。

1995 年，第二版 PHP 问市，定名为 PHP/FI（Form Interpreter）。在这一版本中加入了可以处理更复杂的嵌入式标签语言的解析程序，同时加入了对数据库 MySQL 的支持。自此奠定了 PHP 在动态网页开发上的影响力。自从 PHP 加入了这些强大的功能，它的使用量猛增。据初步统计，

在 1996 年底，有 15 000 个 Web 网站使用了 PHP/FI；而在 1997 年中期，这一数字超过了 50 000。

前两个版本的成功，让 PHP 的设计者和使用者对 PHP 的未来充满了信心。在 1997 年，PHP 开发小组又加入了 Zeev Suraski 及 Andi Gutmans，他们自愿重新编写了底层的解析引擎，另外，还有很多人员也自愿加入了 PHP 其他部分的工作，从此 PHP 成为了真正意义上的开源项目。

1998 年 6 月发布了 PHP 3.0。在这一版本中，PHP 可以跟 Apache 服务器紧密地结合；再加上它的不断更新及加入新的功能；并且它几乎支持所有主流与非主流数据库；而且拥有非常高的执行效率，这些优势使 1999 年使用 PHP 的网站超过了 150 000 个。

经过 3 个版本的演化，PHP 已经变成一个非常强大的 Web 开发语言。这种语言非常易用，而且它拥有一个强大的类库，类库的命名规则也十分规范，使用者就算对一些函数的功能不了解，也可以通过函数名猜出来。PHP 程序可以直接使用 HTML 编辑器来处理，因此，PHP 变得非常流行，有很多大的门户网站都使用了 PHP 作为自己的 Web 开发语言，例如新浪网等。

在 2000 年 5 月推出了划时代的版本 PHP 4。PHP 4 使用了一种“编译-执行”模式，其核心引擎更加优越，提供了更高的性能，而且还包含其他一些关键功能，比如支持更多的 Web 服务器、HTTP Sessions 支持、输出缓存、更安全地处理用户输入的方法以及一些新的语言结构。

2004 年 7 月，PHP 5 发布。该版本以 Zend 引擎Ⅱ为引擎，并且加入了新功能如 PHP Data Objects（PDO）。PHP 5 版本强化更多的功能。首先，完全实现面向对象，提供名为 PHP 兼容模式的功能。其次是 XML 功能，PHP 5 版本支持可直观地访问 XML 数据、名为 SimpleXML 的 XML 处理界面。同时还强化了 XMLWeb 服务支持，而且标准支持 SOAP 扩展模块。数据库方面，PHP 新版本提供旨在访问 MySQL 的新界面——MySQL。除此前的界面外，还可以使用面向对象界面和预处理语句(Prepared Statement)等 MySQL 的新功能. 另外，PHP 5 上还捆绑有小容量 RDBMS-SQLite.

2015 年 6 月，第一版 PHP 7 发布。这是十年来的首次大改版，最大特色是在性能上的大突破，能比前一版 PHP 5 快上一倍。

PHP 目前较新版本是 PHP 7，它在 PHP 5 基础上做了进一步的改进，功能更强大，执行效率更高，性能更强悍。本书将以 PHP 7 版本来讲解 PHP 的实用技能。

1.1.3 PHP 语言的优势

PHP 能够迅速发展，并得到广大使用者的喜爱，主要原因是 PHP 不仅有一般脚本所具有的功能，而且有它自身的优势，具体特点如下。

- 源代码完全开放：事实上，所有的 PHP 源代码都可以获得。读者可以通过 Internet 获得需要的源代码，快速修改并利用。
- 完全免费：和其他技术相比，PHP 本身是免费的。读者使用 PHP 进行 Web 开发无须支付任何费用。
- 语法结构简单：因为 PHP 结合了 C 语言和 Perl 语言的特色，编写简单，方便易懂。可以嵌入于 HTML 语言，实用性强，更适合初学者。
- 跨平台性强：由于 PHP 是运行在服务器端的脚本，可以运行在 UNIX、Linux 和 Windows 下。
- 效率高：PHP 消耗相当少的系统资源，并且程序开发快，运行快。
- 强大的数据库支持：支持目前所有的主流和非主流数据库，使 PHP 的应用对象非常广泛。
- 面向对象：在 PHP 7 中，面向对象方面都有了很大的改进，现在 PHP 完全可以用来开发

大型商业程序。

1.2　PHP 的应用领域

初学者也许会有疑问，PHP 到底能干什么呢？下面将介绍 PHP 的应用领域。

PHP 在 Web 开发方面的功能非常强大，可以完成一款服务器所能完成的工作。有了 PHP，用户可以轻松地进行 Web 开发。下面来具体学习一下 PHP 的应用领域，例如生成动态网页、收集表单数据和发送或接受 Cookies 等。

PHP 主要应用于以下 3 个领域。

1. 服务端脚本

PHP 最主要的应用领域是服务器端脚本。服务器脚本运行需要具备 3 项配置：PHP 解析器、Web 浏览器和 Web 服务器。在 Web 服务器运行时，安装并配置 PHP，然后用 Web 浏览器访问 PHP 程序输出。在学习的过程中，读者只要在本机上配置 Web 服务器，即可浏览制作的 PHP 页面。

2. 命令行脚本

命令行脚本和服务端脚本不同，编写的命令行脚本并不需要任何服务器或浏览器运行，在命令行脚本模式下，只需要 PHP 解析器执行即可。这些脚本在 Windows 和 Linux 平台下作为日常运行脚本，也可以用来处理简单的文本。

3. 编写桌面应用程序

PHP 在桌面应用程序的开发中并不常用，但是如果用户希望在客户端应用程序中使用 PHP 编写图形界面应用程序，可以通过 PHP-GTK 来编写。PHP-GTK 是 PHP 的扩展，并不包含在标准的开发包中，开发用户需要单独编译它。

1.3　PHP 7 的新特点

PHP 7 是 PHP 编程语言的一个主要版本，是开发 Web 应用程序的一次革命，可开发和交付移动企业和云应用。此版本被认为是 PHP 5 后最重要的变化。

和早期版本相比，PHP 7 有以下新的特点。

1. 标量类型声明

PHP 7 增加了对返回类型声明的支持。返回类型声明指明了函数返回值的类型。可用的类型与参数声明中可用的类型相同。例如以下代码：

```
<?php
function arraysSum(array ...$arrays): array
{
return array_map(function(array $array): int {
return array_sum($array);
}, $arrays);
}
```

```
print_r(arraysSum([1,2,3], [4,5,6], [7,8,9]));
?>
```

以上例子会输出：

```
Array
(
[0] => 6
[1] => 15
[2] => 24
)
```

2. null 合并运算符

新增了 null 合并运算符“？？”，它可以替换三元表达式和 isset()。例如以下代码：

```
$a = isset($_GET['a']) ? $_GET['a'] : 1;
```

可以用 null 合并运算符替换如下：

```
$a = $_GET['a'] ?? 1;
```

这两个语句的含义都是：如果变量 a 存在且值不为 NULL，它就会返回自身的值，否则返回它的第二个操作数。可见，新增的??运算符可以简化判断语句。

3. 组合比较符

组合比较符<=>用于比较两个表达式。例如$a<=>$b，表示当$a 大于、等于或小于$b 时它分别返回 1、0 或-1。例如以下代码：

```
<?php
//整型举例
echo 1 <=> 1; //输出 0
echo 1 <=> 2; //输出-1
echo 2 <=> 1; // 输出 1
// 浮点型举例
echo 5.5 <=> 5.5 //输出 0
echo 5.5 <=> 7.0; //输出-1
echo 7.0 <=> 5.5; //输出 1
// 字符串型举例
echo "a" <=> "a"; //输出 0
echo "a" <=> "b"; //输出-1
echo "b" <=> "a"; //输出 1
?>
```

4. 通过 define() 定义常量数组

对于常量数组，可以使用 define()定义，例如以下代码：

```
<?php
define('PERSON', ['xiaoming', 'xiaoli', 'xiaolan']);
echo PERSON[1]; // 输出 "xiaoli"
?>
```

5. 匿名类

现在支持通过 new class 来实例化一个匿名类，这可以用来替代一些“用后即焚”的完整类定义。

6. 支持 Unicode 字符格式

PHP 7 支持任何有效的 codepoint 编码，输出为 UTF-8 编码格式的字符串。例如以下代码：

```
<?php
 echo "\u{6666}";
?>
```

在 PHP 7 环境下输出为：晦，而在早期版本中则输出为：\u{6666}

7. 更多的 Error 变为可捕获的 Exception

PHP 7 实现了一个全局的 throwable 接口，原来的 Exception 和部分 Error 都实现了这个接口（interface），以接口的方式定义了异常的继承结构。于是，PHP 7 中更多的 Error 变为可捕获的 Exception 返回给开发者。如果不进行捕获则为 Error，如果捕获就变为一个可在程序内处理的 Exception。这些可被捕获的 Error 通常都是不会对程序造成致命伤害的 Error，例如函数不存在。PHP 7 进一步方便开发者处理，让开发者对程序的掌控能力更强。因为在默认情况下，Error 会直接导致程序中断，而 PHP 7 则提供捕获并且处理的能力，让程序继续执行下去，为程序员提供更灵活的选择。

例如，执行一个不确定是否存在的函数，PHP 5 兼容的做法是在函数被调用之前追加的判断 function_exist，而 PHP 7 则支持捕获 Exception 的处理方式。

8. 性能大幅度提升

PHP 7 较 PHP 5 相比，速度快 2 倍以上。另外，PHP 7 降低内存消耗，优化后 PHP 7 使用较少的资源，比 PHP 5.6 低了 50%的内存消耗。同时，PHP 7 也支持 64 位架构机器，运算速度更快。PHP 7 可以服务于更多的并发用户，无须任何额外的硬件。

1.4 PHP 常用开发工具

可以编写 PHP 代码的工具有很多，目前常见的有 Dreamweaver、PHPEdit、PHPed 和 Frontpage 等，甚至用 Word 和记事本等常用工具也可以编写源代码。

1.4.1 PHP 代码开发工具

常见的 PHP 代码开发工具有 PHPEdit、gPHPedit、phpDesigner 和 Zend Studio。

1. PHPEdit

PHPEdit 是 Windows 下一款优秀的 PHP 脚本 IDE（集成开发环境）。该软件为快速、便捷地开发 PHP 脚本提供了多种工具，其功能包括：语法关键词高亮，代码提示、浏览，集成 PHP 调试工具，帮助生成器，自定义快捷方式，150 多个脚本命令，键盘模板，报告生成器，快速标记和插

件等。

2. gPHPedit

gPHPedit 是 Linux 下十分流行的免费的 PHP 编辑器，它小巧而功能强大。它以 Linux 下的 gedit 文本编辑器为基础，是专门用于编辑 PHP 和 html 的编辑器。它可以显著表示 PHP、html、css 以及 SQL 语句。在编写过程中提供函数列表参考、函数参数参考、搜索和检测编程语法等。总之，gPHPedit 是一款完全免费的优秀的 PHP 编辑器。

3. PhpDesigner

PhpDesigner 是一款功能强大的、运行高效的、优秀的 PHP 编辑平台。它集合了 PHP、xhtml、JavaScript、css 等基于 Web 开发的综合 Web 应用开发平台。它能够自动捕获代码文件中的 class、function、varibles 等编程元素，并加以整理，在编程过程中给予提示。除此以外，它还兼容了流行的各种类库和框架的协同工作，比如 JavaScript 的 jQuery 库、YUI 库、prototype 库等，又比如 PHP 流行的 zend framework 框架、symfony 框架、cakephp 框架、yii 框架等。此外，它还拥有 xdebug、svn 版本管理等工具。可以说它是独立于 eclipse 之外的，集合了 PHP 开发需求之大成的又一款优秀的平台。

4. Zend Studio

Zend Studio 是由 zend 科技开发的一个针对 PHP 的全面开发平台。这个 IDE 融合了 zend server 和 zend framework，并且融合了 Eclipse 开发环境。Eclipse 是最早适用于 Java 的 IDE 环境，但是由于其优良的特性和对 PHP 的支持，成为很有影响力的 PHP 开发工具。现在，最新的 Eclipse PHP 开发环境为 Eclipse PDT 2.2.0 版本，它拥有支持 Windows、Linux 和 Mac 系统的软件包，可以说它拥有比较完备的体系。它是一款收费的工具。

1.4.2 网页设计工具

下面介绍两种常见的网页设计工具。

1. Dreamweaver

Dreamweaver 是网页制作“三剑客”之一，其功能更多体现在对 Web 页面 HTML 的设计上。随着 Web 语言的发展，Dreamwaver 的功能早已不再仅限于网页设计的方面，它更多地支持各种 Web 应用流行的前后台技术的综合运用。Dreamwaever 对 PHP 的支持十分到位，它不但对 PHP 的不同方面进行清晰地表示，并且给予足够的编程提示，使编程过程相当流畅。

2. FrontPage

FrontPage 是微软公司出品的一款网页制作入门级软件。FrontPage 使用方便简单，会用 Word 就能做网页，所见即所得是其特点，该软件结合了设计、拆分、代码和预览 4 种模式。

1.4.3 文本编辑工具

常见的文本编辑工具有很多，包括 UitraEdit 和记事本等。

1. UitraEdit

UltraEdit 是一套功能强大的文本编辑器，可以编辑文本、十六进制、ASCII 码，完全可以取代记事本（如果电脑配置足够强大），内建英文单词检查、C++ 及 VB 指令突显，可同时编辑多个文件，而且即使开启很大的文件速度也不会慢。软件附有 HTML 标签颜色显示、搜寻替换以及无限制的还原功能，一般用其来修改 EXE 或 DLL 文件。UitraEdit 是能够满足用户一切编辑需要的编辑器。

2. 记事本

记事本是 Windows 系统自带的文本编辑工具。它具备最基本的文本编辑功能，体积小巧，启动快，占用内存少，容易使用。记事本主窗口如图 1-1 所示。

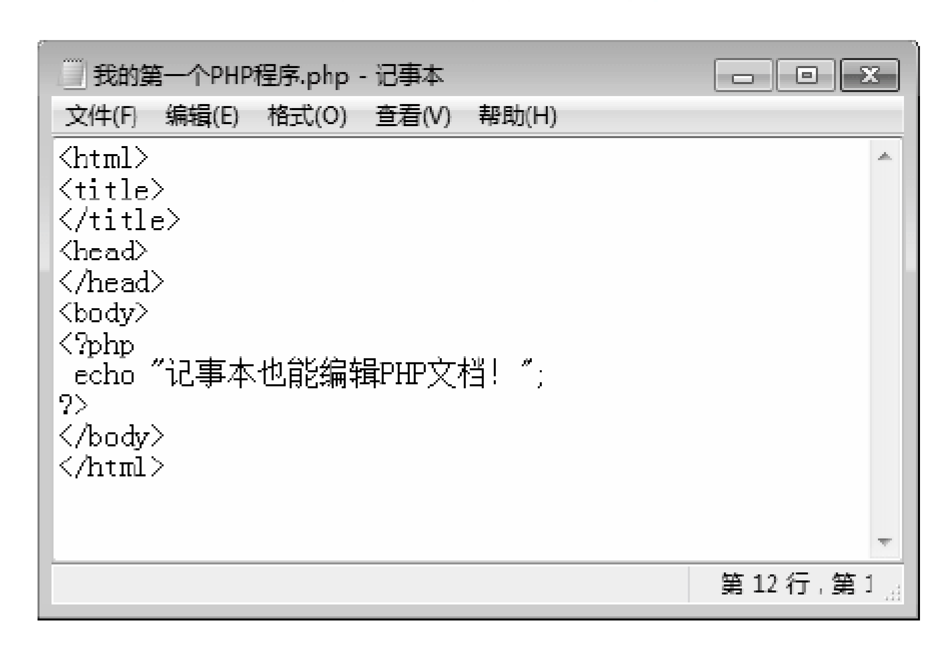


图 1-1 记事本主窗口

在使用记事本编辑 PHP 文档的过程中，需要注意保存方法和技巧。在【另存为】对话框中输入文件名称，后缀名为.php。另外，将【保存类型】设置为【所有文件】即可，如图 1-2 所示。

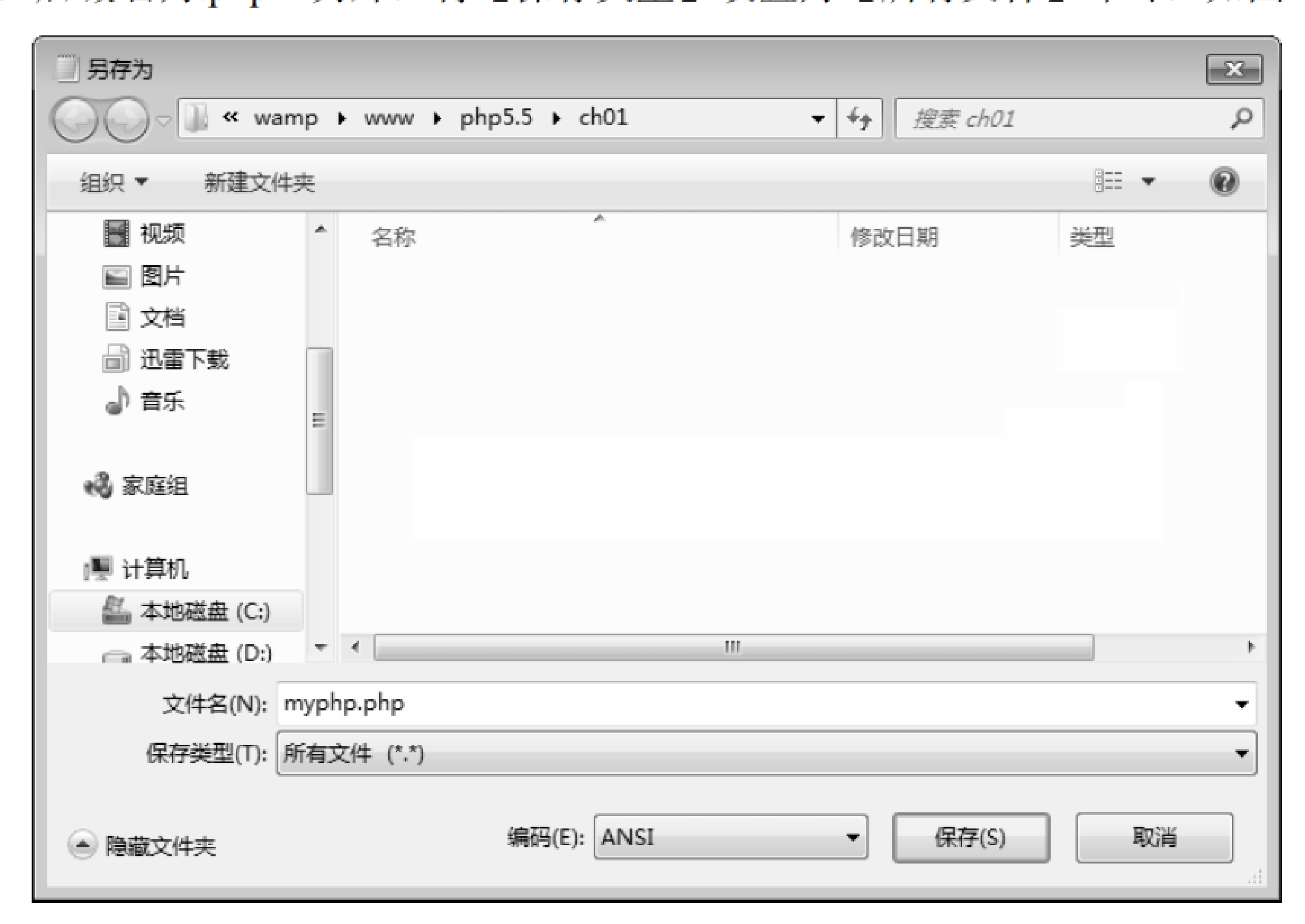


图 1-2 【另存为】对话框

1.5 高手私房菜

技巧 1：如何快速了解 PHP 的应用技术？

在学习的过程中，用户可以随时查阅 PHP 的应用情况。启动 IE 浏览器，在地址栏中输入 http://www.baidu.com，打开搜索引擎，输入需要搜索的内容即可了解相关技术。

技巧 2：如何选择 PHP 开发软件？

无论是哪个开发工具，在 PHP 的开发过程中，都要有对 PHP 的语法和数据进行分色表示的能力，以方便开发者编写程序。进一步的功能是要有对代码编写提示的能力，对于 PHP 的数据类型、运算符、标识、名称等都有提示功能。

那么多的开发工具，选择一款比较适合自己的即可。就初学者而言，使用 phpDesigner 比较好，它集合了 PHP、XHTML、JavaScript、CSS 等基于 Web 开发的综合 Web 应用开发技术，对于学习 Web 开发的初学者而言，随时可以学习相关的 Web 技术。

1.6 经典习题

（1） 查询 PHP 的应用领域。

（2） 在百度中查询 PHP 7 的新功能和优势。

第 2 章　配置 PHP 7 服务器环境

在编写 PHP 文件之前，读者需要配置 PHP 服务器，包括软硬件环境的检查、如何获得 PHP 安装资源包等，本章将详细讲解目前常见的主流 PHP 服务器搭配方案：PHP 7+IIS 和 PHP 7+Apache。另外将讲述在 Windows 下如何使用 WampServe 组合包，最后通过一个实战演练，检查 Web 服务器的构建是否成功。

本章学习目标

- 了解 PHP 服务器的概念
- 熟悉安装 PHP 7 前的准备工作
- 掌握 PHP 7+Apache 服务器的环境搭建方法

2.1　PHP 服务器概述

在学习 PHP 服务器之前，读者需要了解 HTML 网页的运行原理。网页浏览者在客户端通过浏览器向服务器发出页面请求，服务器接收到请求后将页面返回到客户端的浏览器，这样网页浏览者即可看到页面显示效果。

PHP 语言在 Web 开发中作为嵌入式语言，需要嵌入到 HTML 代码中执行。要想运行 PHP 网站，需要搭建 PHP 服务器。PHP 网站的运行原理如图 2-1 所示。

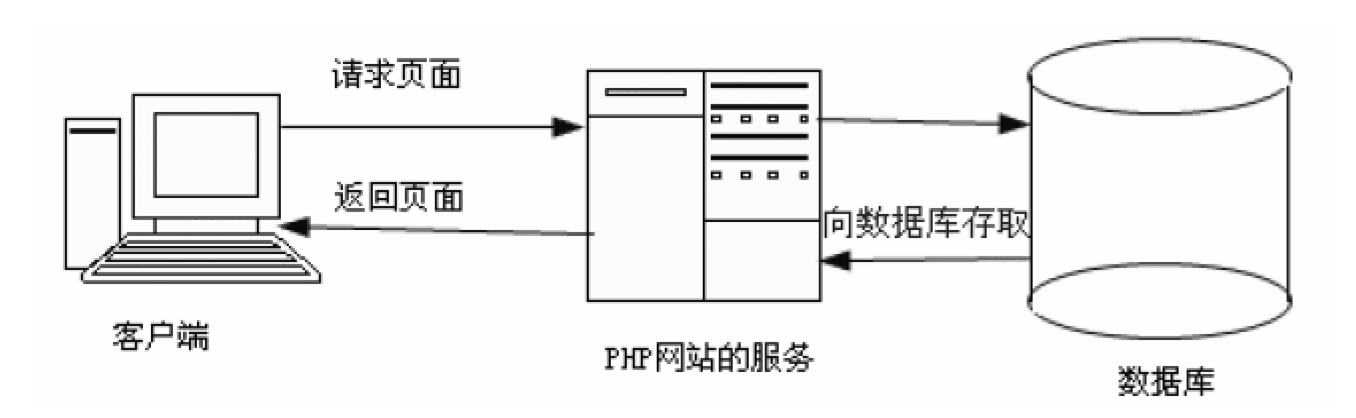


图 2-1　PHP 网站运行流程图

从图 2-1 可以看出，PHP 程序运行的基本流程如下。

- 网页浏览者首先在浏览器的地址栏中输入要访问的主页地址，按回车键触发这个申请。
- 浏览器将申请发送到 PHP 网站服务器。网站服务器根据申请读取数据库中的页面。
- 通过 Web 服务器向客户端发送处理结果，客户端的浏览器显示最终页面。



提　示

由于在客户端显示的只是服务器端处理过的 HTML 代码页面，所以网页浏览者看不到 PHP 代码，这样可以提高代码的安全性。同时在客户端不需要配置 PHP 的环境，主要安装浏览器即可。

2.2 安装 PHP 7 前的准备工作

在安装 PHP 之前，要了解安装所需要的软硬件环境和如何获取 PHP 安装资源包。

2.2.1 软硬件环境

大部分软件在安装过程中都需要软硬件环境的支持，当然 PHP 也不例外。在硬件方面，如果只是为了学习上的需求，PHP 只需要一台普通的电脑即可。在软件方面需要根据实际工作的需要选择不同的 Web 服务软件。

PHP 具有跨平台特性，所以 PHP 开发用什么样的系统不太重要，开发出来的程序都能很轻松地移植到其他操作系统中。另外，PHP 开发平台支持目前主流的操作系统，包括 Windows 系列、Linux、Unix 和 Mac OS X 等。下面以 Windows 7 平台为例进行讲解。

另外，用户还需要安装 Web 服务软件。目前，PHP 支持大多数 Web 服务软件，常见的有 IIS、Apache、PWS 和 Netscape 等。比较流行的是 IIS 和 Apache，下面将详细讲述这两种 Web 服务器的安装和配置方法。

2.2.2 获取 PHP 7 安装资源包

PHP 安装资源包中包括安装和配置 PHP 服务器的所需文件和 PHP 扩展函数库。获取 PHP 安装资源包的方法比较多，很多网站都提供 PHP 安装包，建议读者从官方网站下载，具体操作步骤如下。

01 打开 IE 浏览器，在地址栏中输入下载地址 http://windows.php.net/download，按回车键确认，进入 PHP 下载网站，如图 2-2 所示。

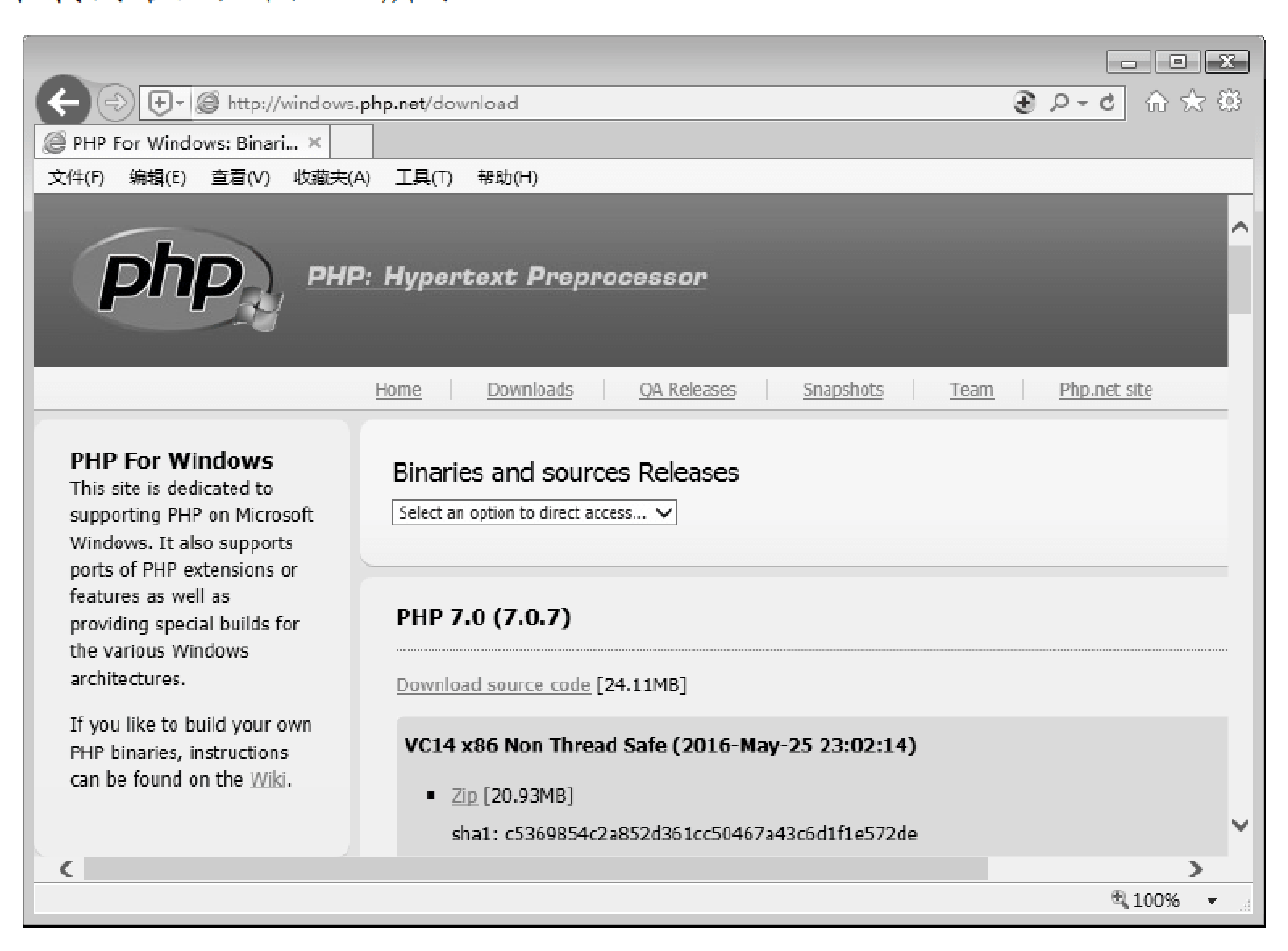


图 2-2 PHP 网站下载页面

02 进入下载页面，单击【Binaries and sources Releases】下方的下拉菜单，在弹出的下拉列表中选择适合的版本，这里选择 PHP 7 版本，如图 2-3 所示。

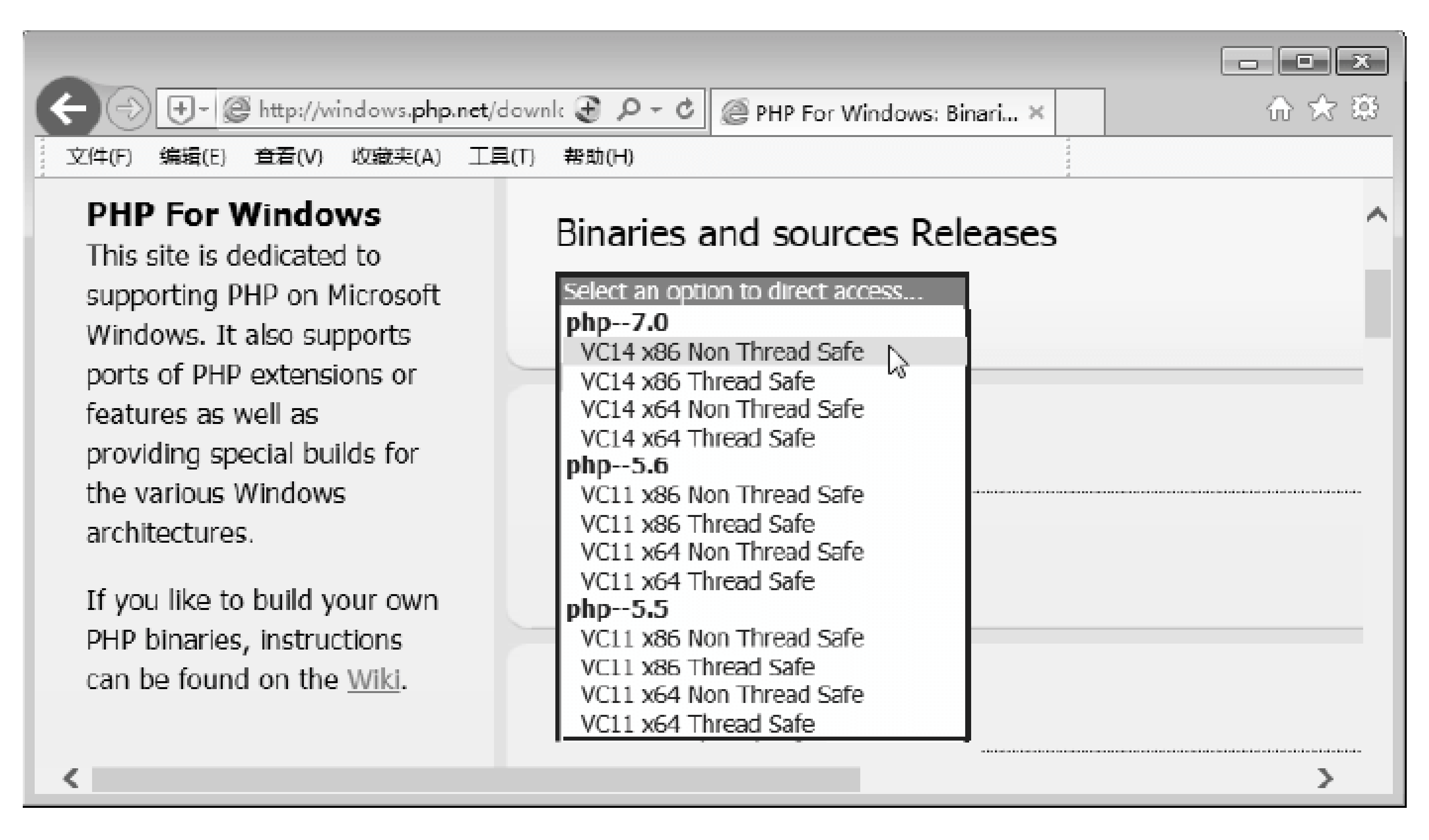


图 2-3 选择需要的版本



提 示

在图 2-3 的下拉列表中 VC14 代表的是 Visual Studio 2015 Compiler 编译器编译，通常使用在 IIS+PHP 服务器下，要求用户安装 Visual C++ Redistributable for Visual Studio 2015。

03 显示所选版本号中 PHP 安装包的各种格式，这里选择 Zip 的压缩格式，单击【Zip】链接，如图 2-4 所示。

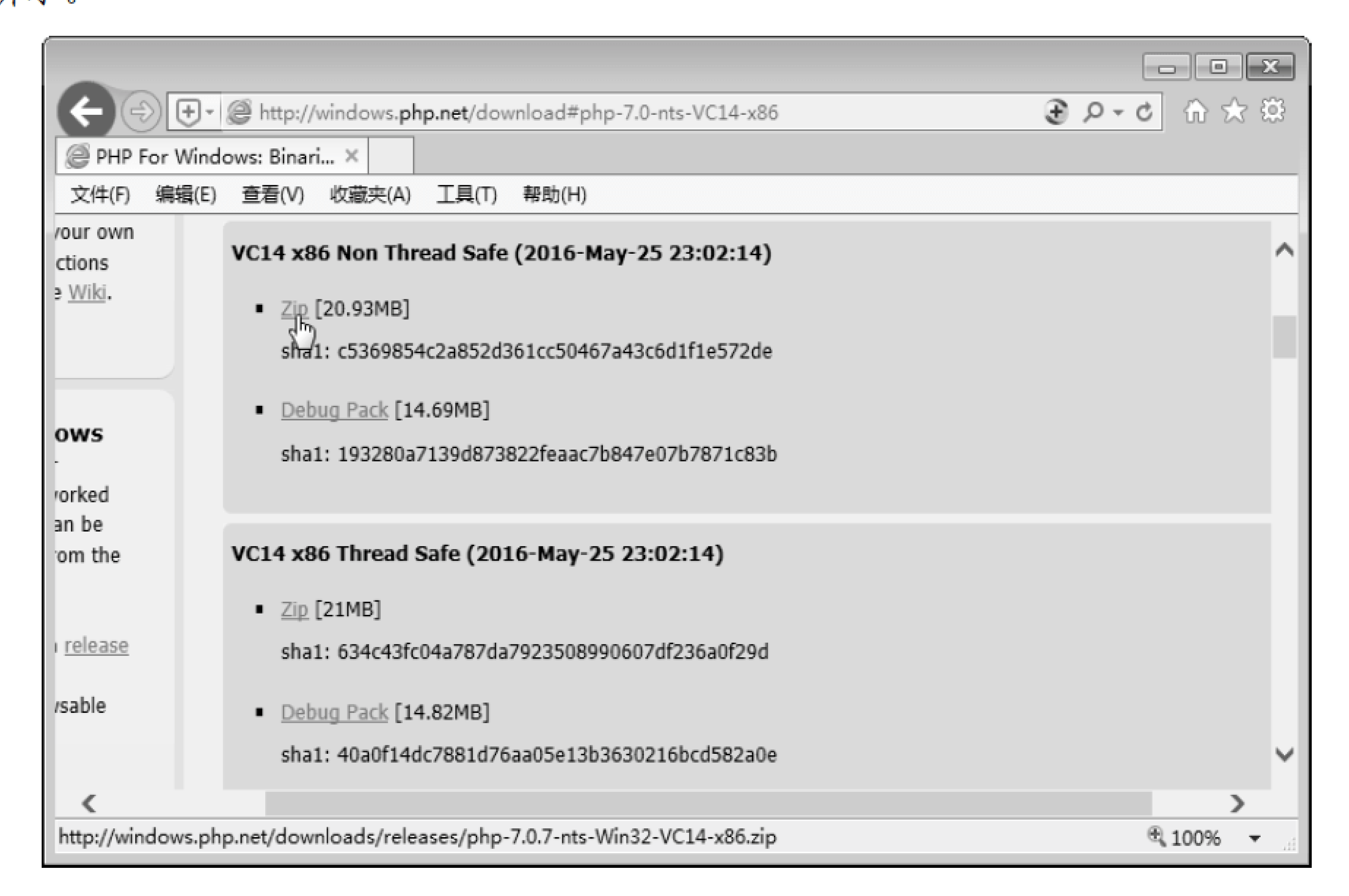


图 2-4 选择需要版本的格式

04 打开【另存为】对话框，选择保存路径，然后保存文件即可，如图 2-5 所示。

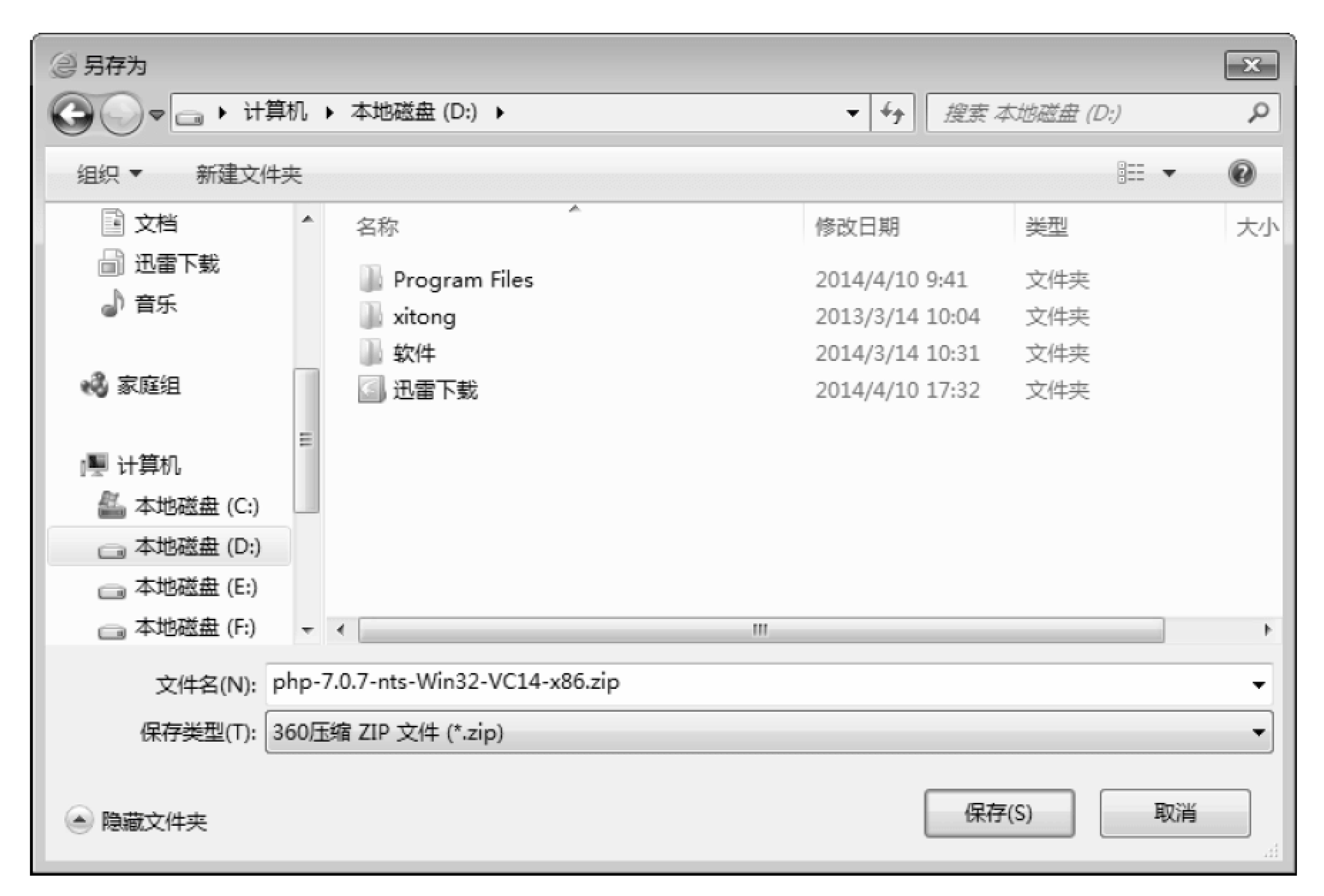


图 2-5 【另存为】对话框

2.3 PHP 7+Apache 服务器的环境搭建

Apache 支持大部分操作系统，搭配 PHP 程序的应用，就可以开发出功能强大的互动网站。本节主要讲述 PHP 7+Apache 服务器的搭建方法。

2.3.1 Apache 简介

Apache 是世界排名第一的 Web 服务器软件。它可以运行在几乎所有广泛使用的计算机平台上，由于其跨平台特性和安全性被广泛使用，是最流行的 Web 服务器端软件之一。

和一般的 Web 服务器相比，Apache 的主要特点如下。

- 跨平台应用：几乎可以在所有的计算机平台上运行。
- 开发源代码：Apache 服务程序由全世界众多开发者共同维护，并且任何人都可以自由使用，充分体现了开源软件的特性。
- 支持 HTTP/1.1 协议：Apache 是最先使用 HTTP/1.1 协议的 Web 服务器之一，它完全兼容 HTTP/1.1 协议并与 HTTP/1.0 协议向后兼容。Apache 已为新协议所提供的全部内容做好了必要的准备。
- 支持通用网关接口（CGI）：Apache 遵守 CGI/1.1 标准并且提供了扩充的特征，如定制环境变量和很难在其他 Web 服务器中找到的调试支持功能。
- 支持常见的网页编程语言：可支持的网页编程语言包括 Perl、PHP、Python 和 Java 等，支持各种常用的 Web 编程语言，使 Apache 具有更广泛的应用领域。
- 模块化设计：通过标准的模块实现专有的功能，提高了项目完成效率。
- 运行非常稳定，同时具备效率高、成本低的特点，而且具有良好的安全性。

2.3.2 关闭原有的网站服务器

在安装 Apache 网站服务器之前，如果所使用的操作系统已经安装了网站服务器，如 IIS 网站服务器等，用户必须先停止这些服务器，才能正确安装 Apache 网站服务器。

以 Windows 7 的操作系统为例，请在桌面上右击【我的电脑】图标，在弹出的快捷菜单中选择【管理】菜单命令，打开【计算机管理】窗口，在左侧的列表中展开【服务和应用程序】选项，然后选择【Internet 信息服务（IIS）管理器】选项，在右侧的列表中单击【停止】按钮即可停止 IIS 服务器，如图 2-6 所示。

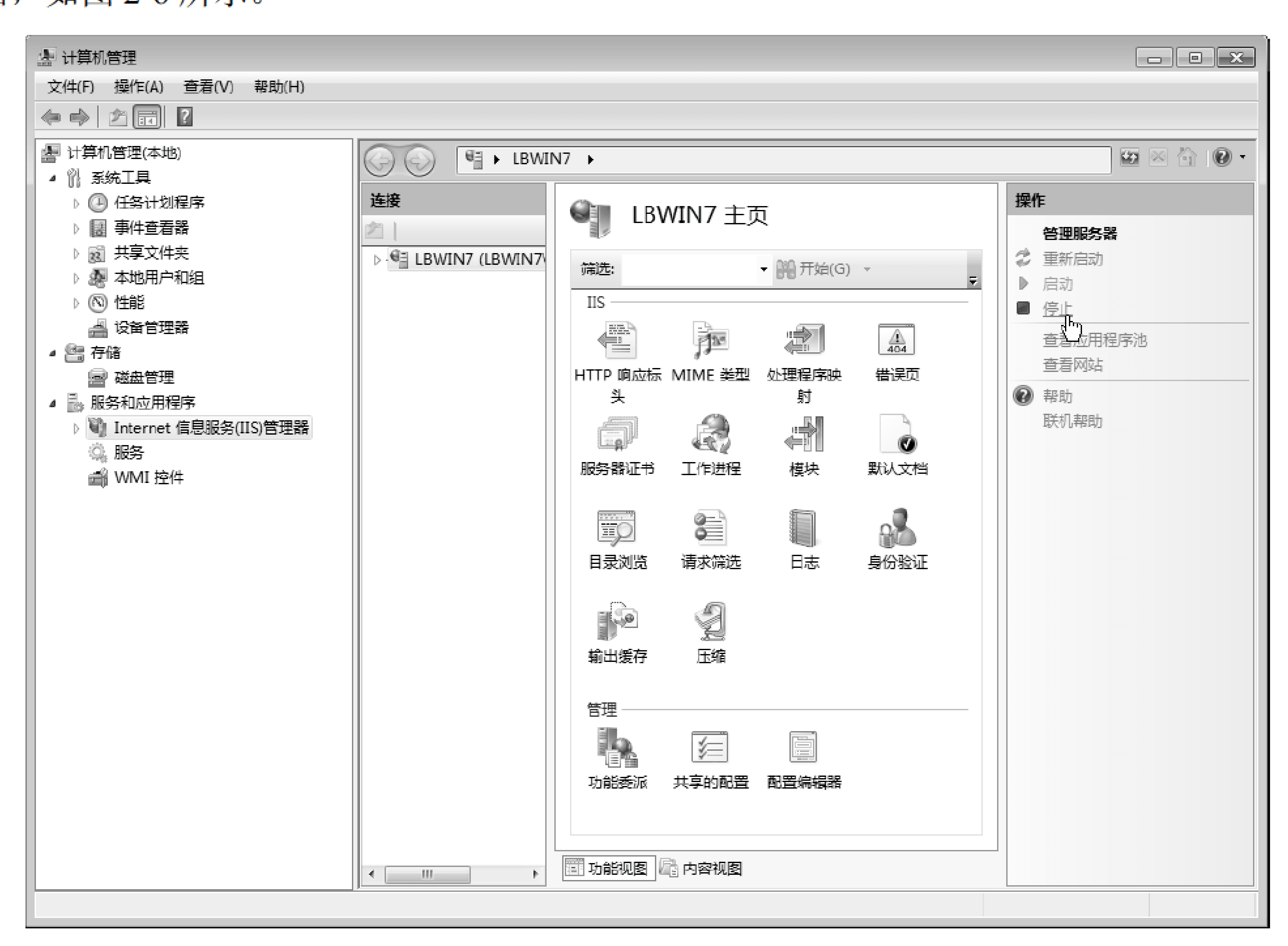


图 2-6 【计算机管理】窗口

如此一来，原来的服务器软件立即失效不再工作，也不会与 Apache 网站服务器产生冲突。当然，如果用户的系统原来就没有安装 IIS 等服务器软件，即可略过这一节的步骤直接往下执行。

2.3.3 安装 Apache

Apache 是免费软件，用户可以从官方网站直接下载。Apache 的官方网站为：http://www.apache.org。

下面以下载好的 Apache 2.4 为例，讲解如何安装 Apache，具体操作步骤如下。

01 在浏览器的地址栏中输入 http://windows.php.net/download/，按回车键，进入 Apache 2.4 下载页面，根据系统的位数选择 32 位或者 64 位，这里选择 32 位的 Apache 2.4，如图 2-7 所示。

02 下载完成后，解压到 D 盘中，这里解压路径为：D:\web\apache24\，如图 2-8 所示。

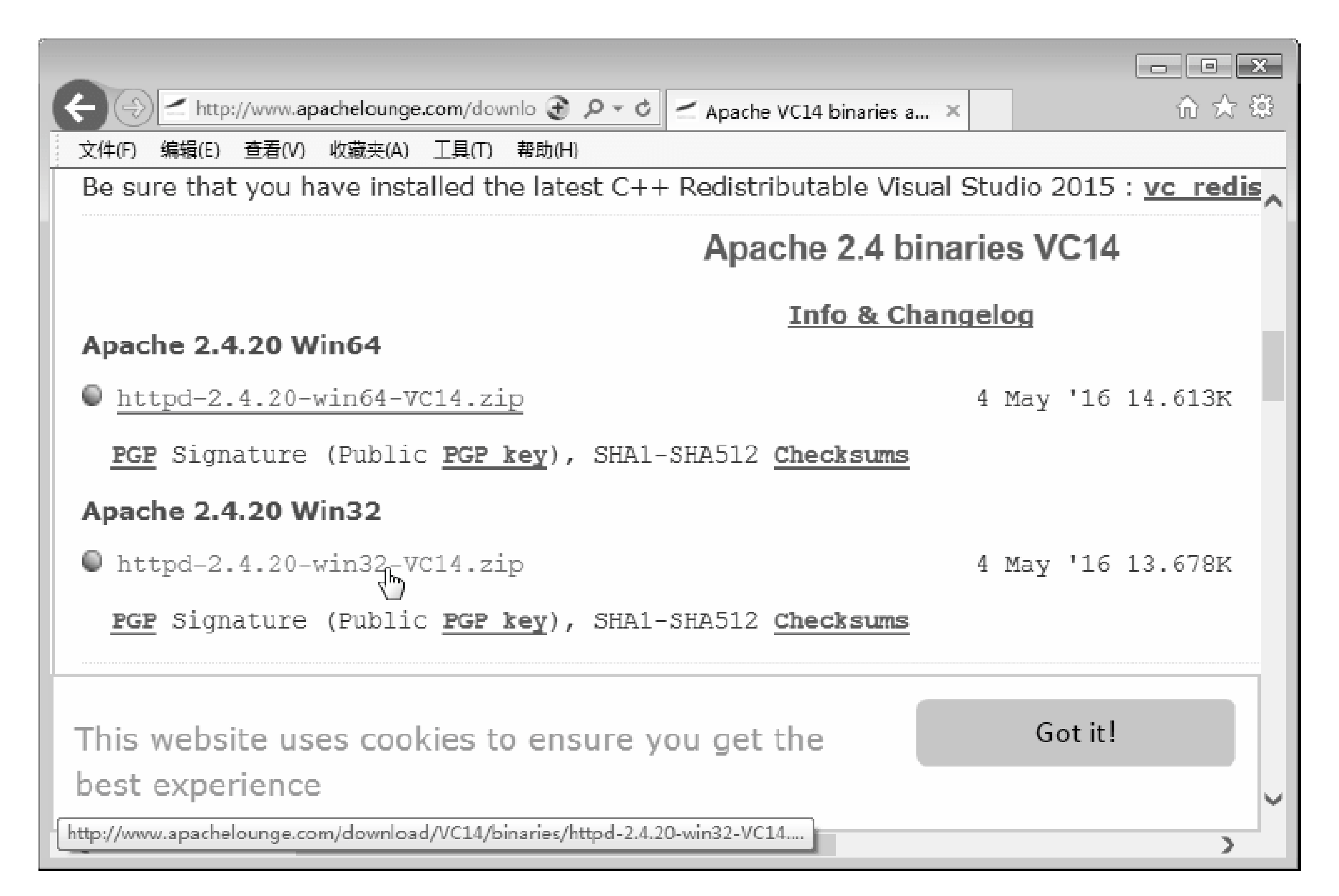


图 2-7　Apache 2.4 下载页面

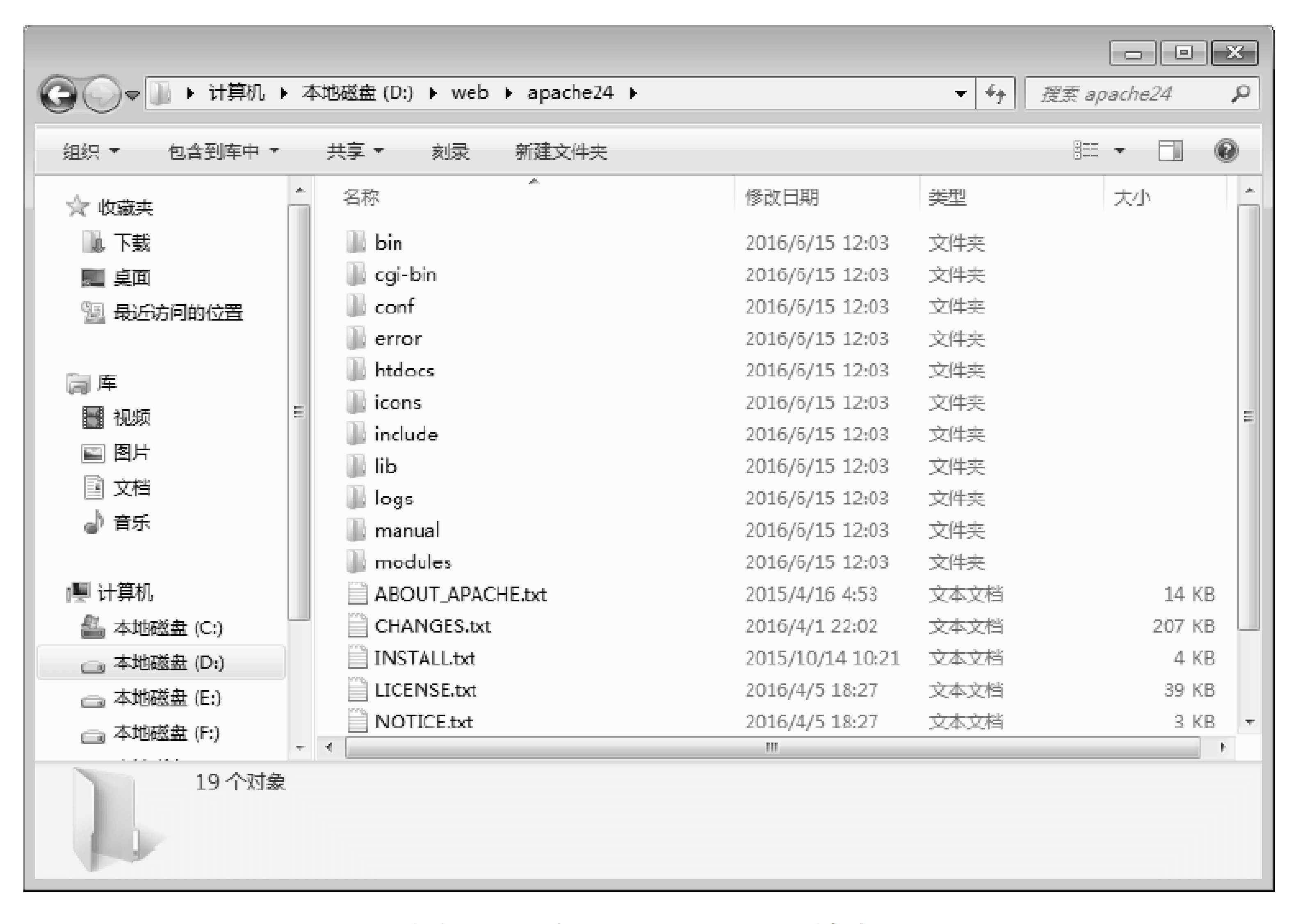


图 2-8　解压 Apache 压缩包

2.3.4　将 PHP 与 Apache 建立关联

Apache 解压完成后，还不能运行 PHP 网页，需要将 PHP 与 Apache 建立关联。

Apache 的配置文件名称为 httpd.conf，它是纯文本文件，用记事本即可打开编辑。此文件存放在 Apache 安装目录的 apache24/conf/下，打开 Apache 的配置文件，首先设置网站的主目录。这里将案例的源文件放在 D 盘的 phpbook 文件夹下，所以设置主目录为 d:/phpbook/。在 http.conf 文件中找到 DocumentRoot 参数，将其值修改为 d:/phpbook/，如图 2-9 所示。

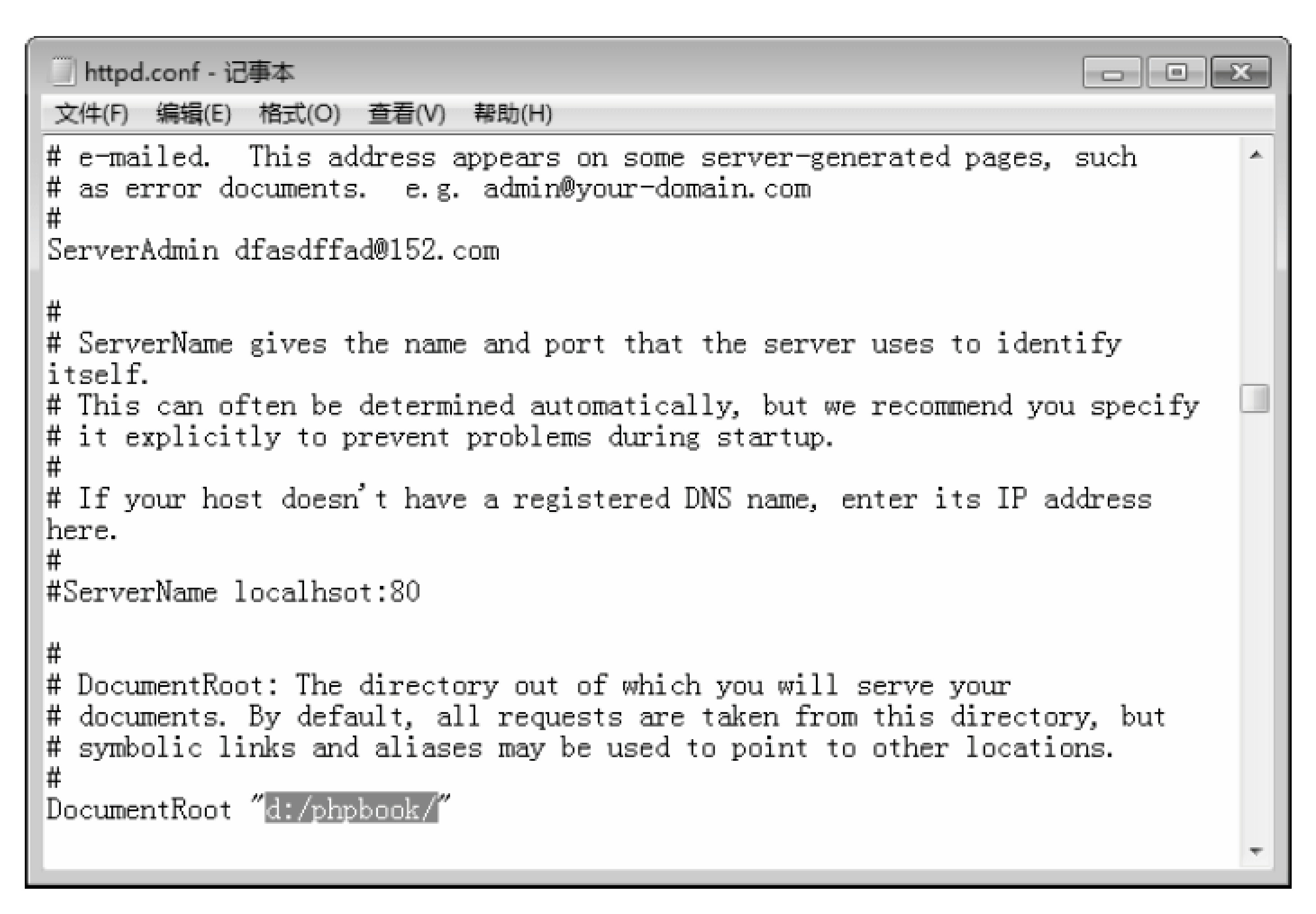


图 2-9　设置网站的主目录

下面指定 php.ini 文件的存放位置。由于 PHP 安装在 d:\PHP 7，所以 php.ini 位置为 d:\PHP 7\php.ini。在 httpd.conf 配置文件中的任意位置加入语句 PHPIniDir "d:\PHP 7\php.ini"，如图 2-10 所示。

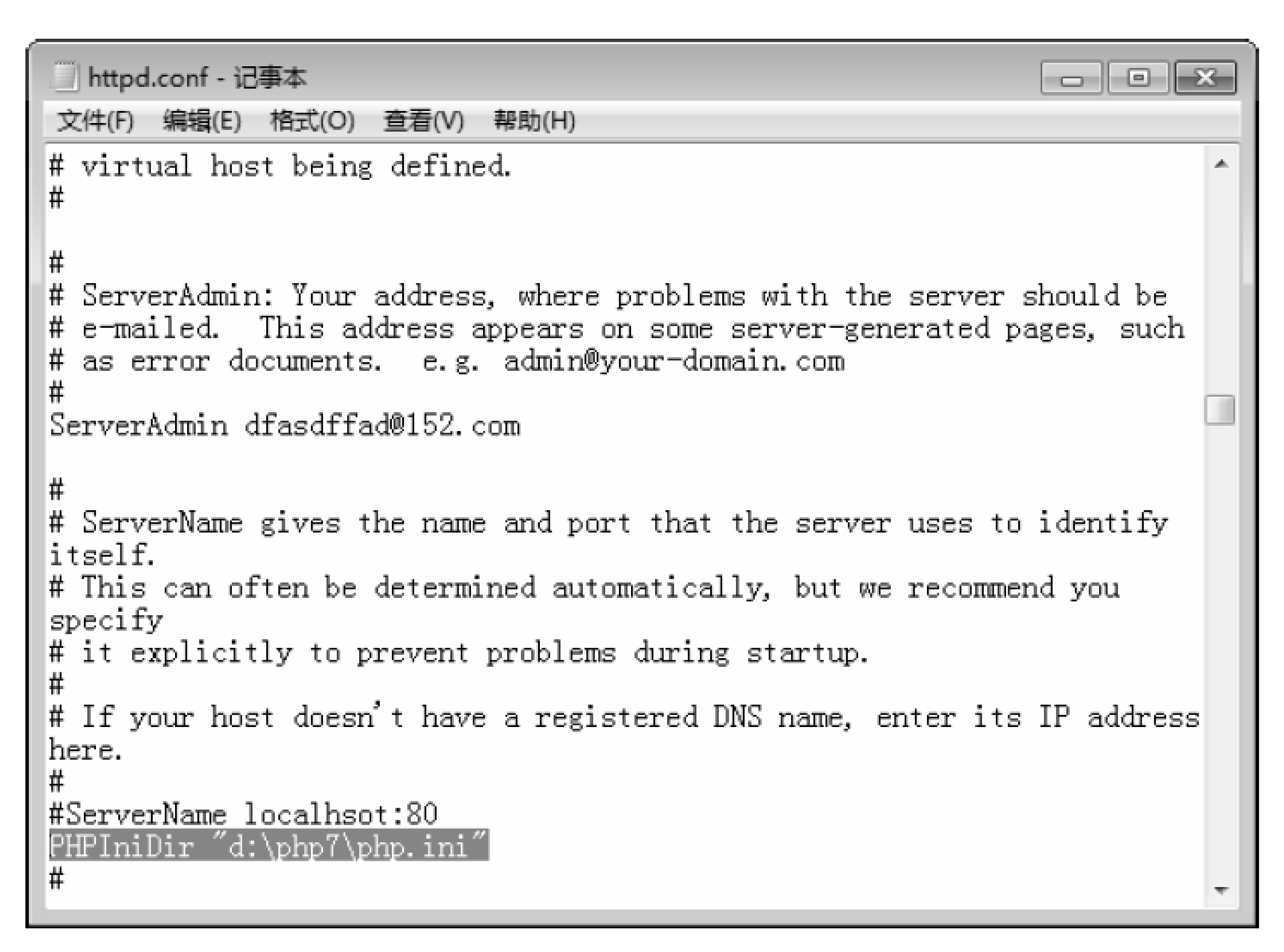


图 2-10　指定 php.ini 文件的存放位置

最后向 Apache 中加入 PHP 模块。在 httpd.conf 配置文件中的任意位置加入 3 行语句：

```
LoadModule PHP 7_module "d:/PHP 7/PHP 7apache2_4.dll"
AddType application/x-httpd-php .php
AddType application/x-httpd-php .html
```

输入效果如图 2-11 所示。完成上述操作后，保存 httpd.conf 文件。



图 2-11　向 Apache 中加入 PHP 模板

最后就是把 apache 加入 windows 服务，并启动 apache。单击【开始】菜单，在弹出的菜单中选择【运行】命令，在【打开】文本框中输入 cmd，单击【确定】按钮，进入 DOS 命令窗口，首先进入 Apache2.4 的目录下，命令如下：

```
cd  D:\web\apache24\
```

启动 apache 服务，命令如下：

```
httpd  -k install
httpd  -k start
```

2.4　PHP 环境的集成软件

对于刚开始学习 PHP 的程序员，往往为了配置环境而不知所措。为此，这里介绍一款对新手非常实用的 PHP 集成开发环境。

WampServe 是指在 Windows 服务器上使用 Apache、MySQL、PHP 和 phpmyadmin 的集成安装环境，目前 WampServe 3 已经支持 PHP 7 版本。由于 WampServe 安装简单、速度较快、运行稳定，所以受到广大初学者的青睐。


提　示

在安装 WampServer 组合包之前，需要确保系统中没有安装 Apache、PHP 和 MySQL。否则，需要先讲这些软件卸载，然后才能安装 WampServer 组合包。

安装 WampServer 组合包具体操作步骤如下：

01 到 wampserver 的官方网站 http://www.wampserver.com/en/下载 wampserver 的最新安装包 WampServer 3 -x32.exe 文件。

02 直接双击安装文件，打开选择安装语言对话框，这里采用默认的设置，单击 OK 按钮，如图 2-12 所示。

03 在打开的窗口中选择【I accept the agreement】单选按钮，如图 2-13 所示。



图 2-12　选择安装语言对话框

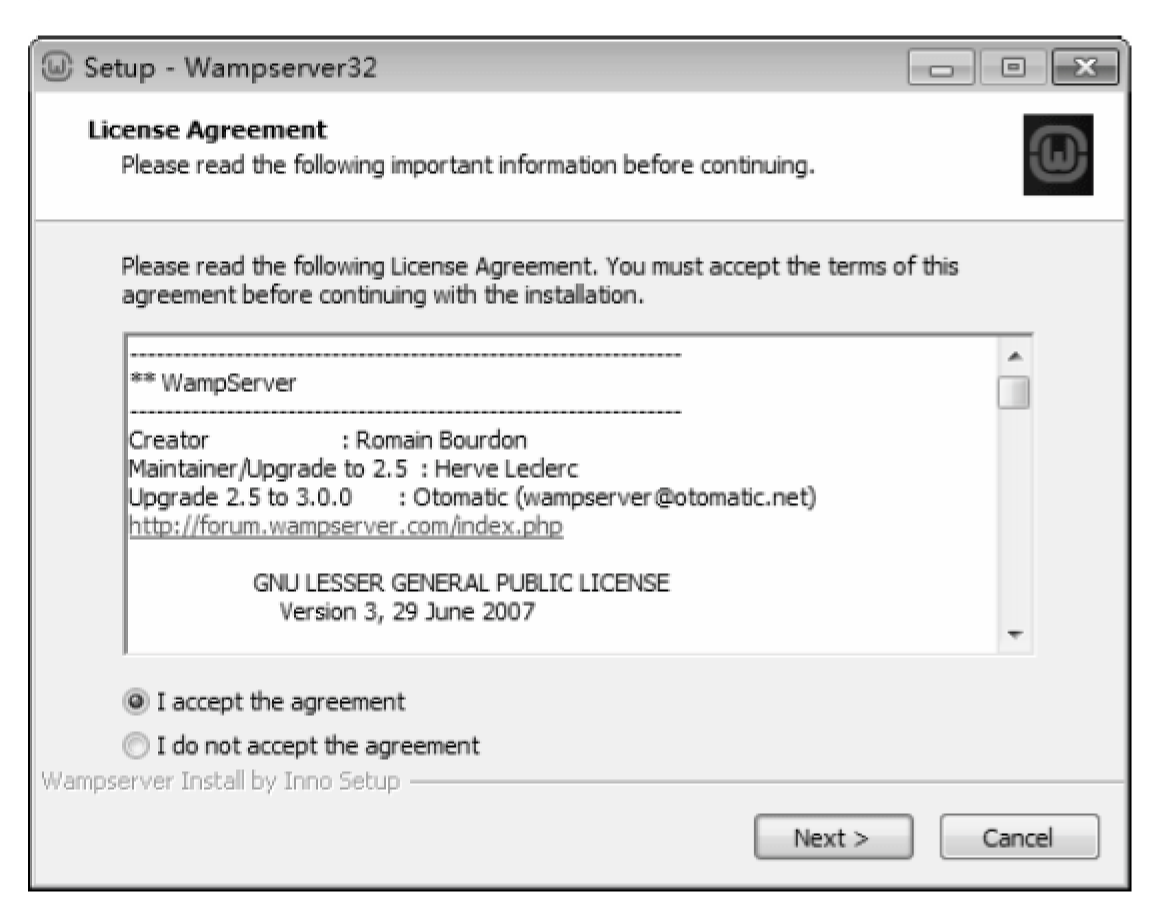
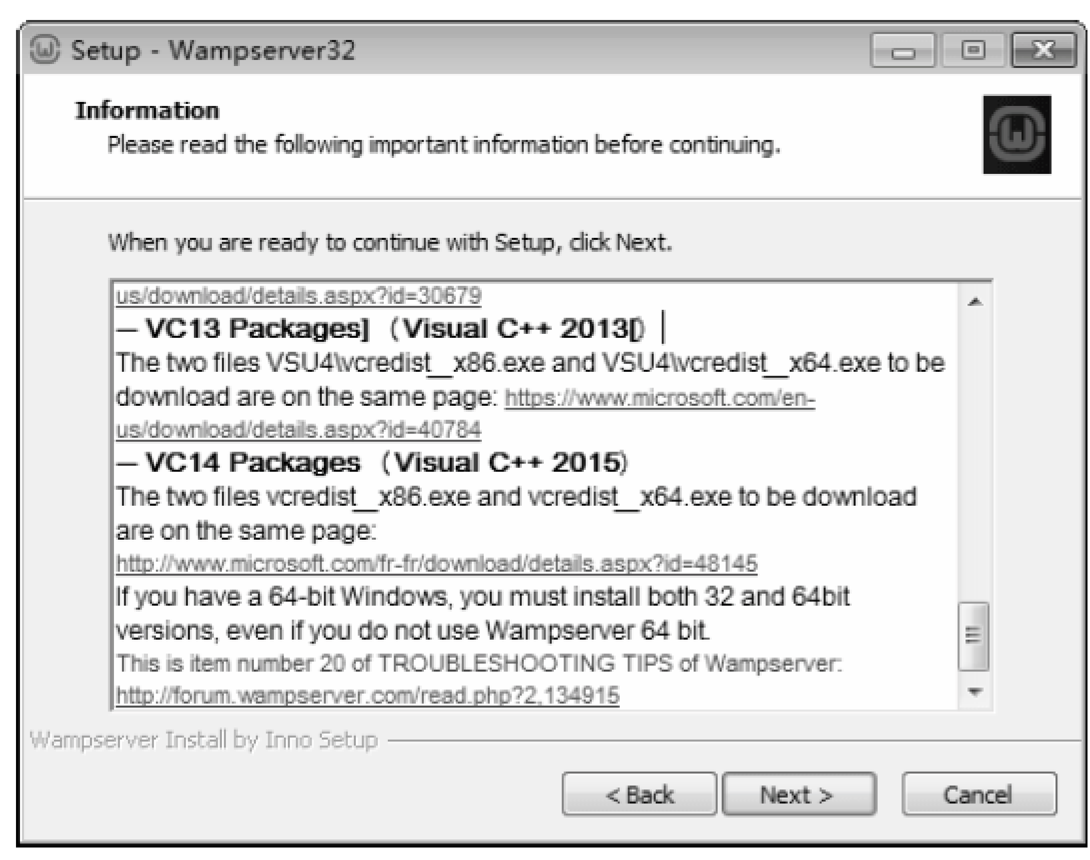


图 2-13　接受许可证协议

04 单击【next】按钮，打开检查安装信息窗口，提示用户需要在系统中安装 Microsoft Visual C++ 2015，否则安装会出错，如图 2-14 所示。

05 单击【next】按钮，在打开的窗口中设置安装路径，这里采用默认设置，如图 2-15 所示。



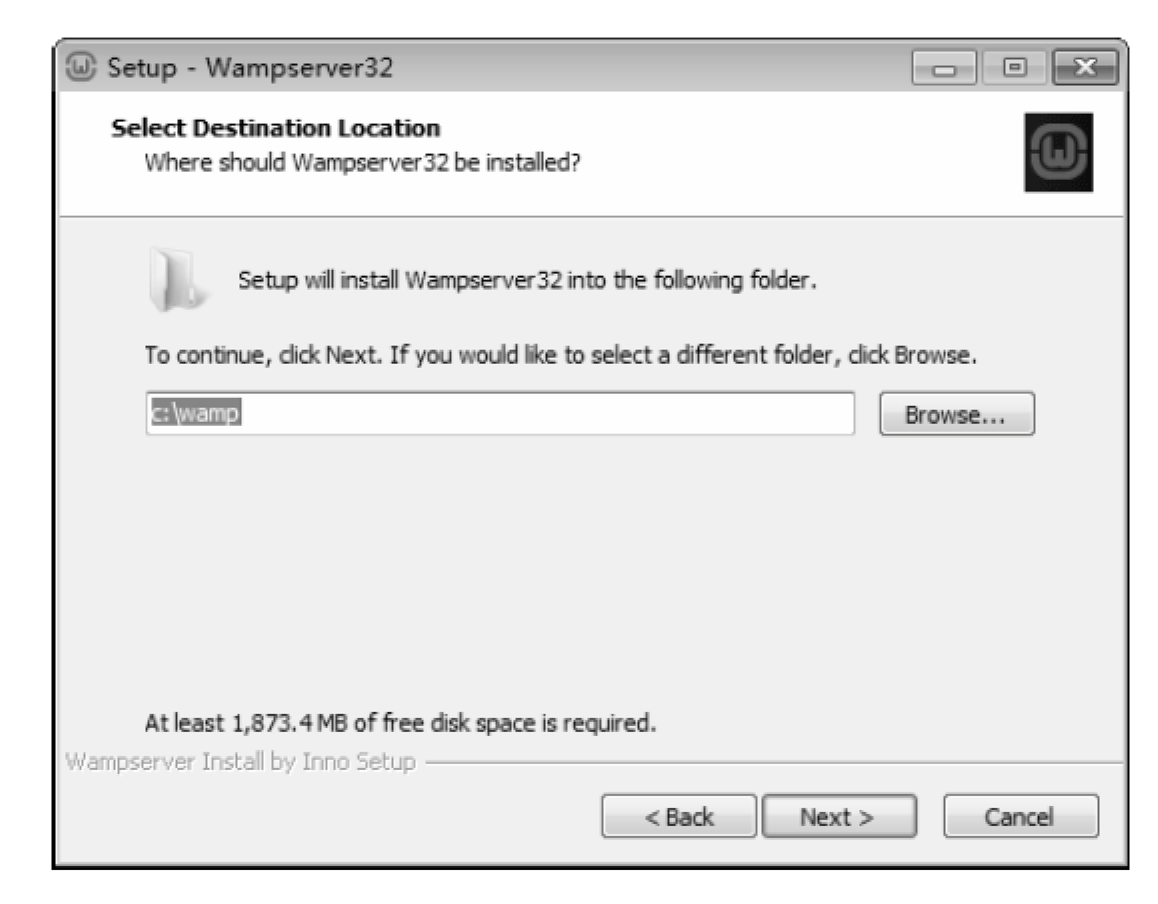


图 2-14　检查安装信息　　　　图 2-15　设置安装路径

06 单击【next】按钮，在打开的窗口中确认安装的参数后，单击【Install】按钮，如图 2-16 所示。

07 程序开始自动安装，并显示安装的进度，如图 2-17 所示。



图 2-16　确认安装窗口

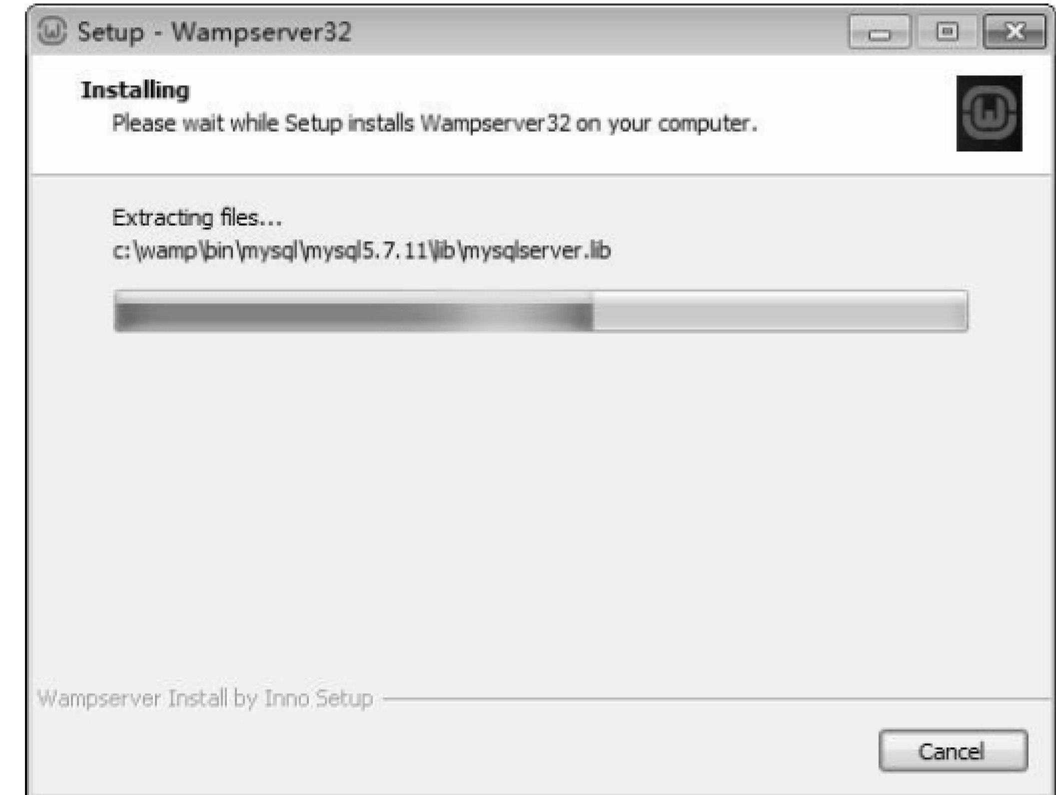


图 2-17　程序开始安装

08 程序安装过程完毕后，进入安装完成窗口，单击【Finish】按钮，完成 WampServer 的安装操作，如图 2-18 所示。

09 单击桌面右侧 WampServer 服务按钮，在弹出的列表中选择【Localhost】选项，如图 2-19 所示。

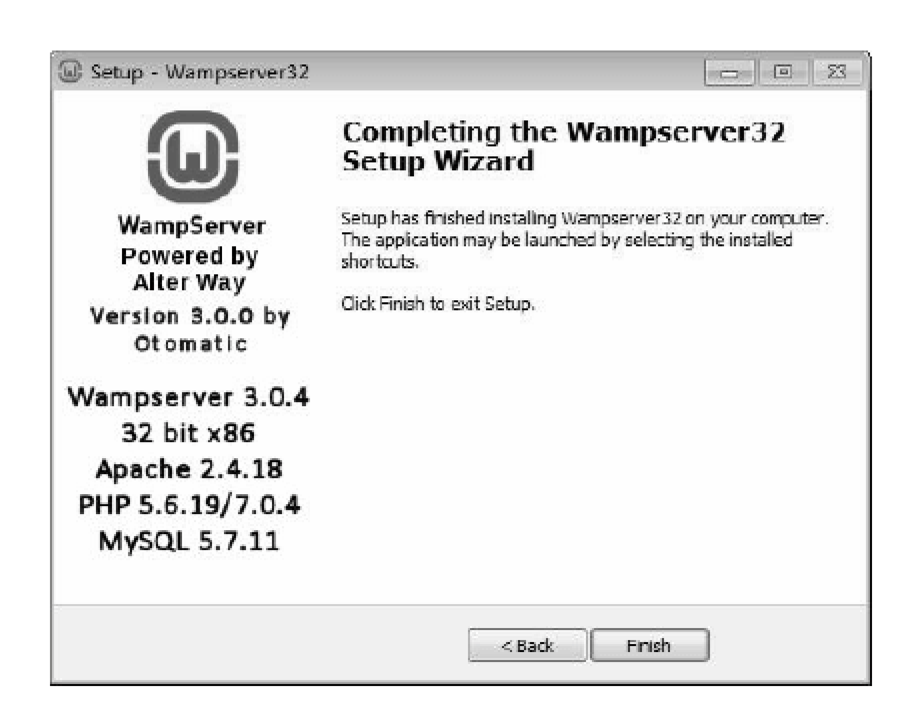


图 2-18　完成安装



图 2-19　WampServer 服务列表

10 自动打开浏览器，显示 PHP 配置环境的相关信息，如图 2-20 所示。



图 2-20　PHP 配置环境的相关信息

2.5 实战演练——我的第一个 PHP 程序

前面讲述了 3 种服务器环境的搭建方法，读者可以根据自己的需求进行选择。

下面通过一个实例讲解如何编写 PHP 程序并运行查看效果。以 IIS 服务器环境为例进行讲解。读者可以使用任意的文本编辑软件，如记事本，新建名称为 hello world 的文件，并输入以下代码。

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<h2>PHP Hello World - 来自 PHP 的问候。</h2>
<?php
  echo "Hello, World.";
  echo "你好世界。";
?>
</BODY>
</HTML>
```

将文件保存在主目录或虚拟目录下，保存格式为.php。在浏览器的地址栏中输入 http://localhost/helloworld.php，并按回车键确认，运行结果如图 2-21 所示。



图 2-21 运行结果

【案例分析】：

（1）其中“PHP Hello World - 来自 PHP 的问候。”是 HTML 中的“<h2>PHP Hello World - 来自 PHP 的问候。</h2>”所生成的。

（2）“Hello, World.你好世界。”则是由“<?php echo "Hello, World."; echo "你好世界。"; ?>”生成的。

（3）在 HTML 中嵌入 PHP 代码的方法即是在<?php ?>标识符中间填入 PHP 语句，语句要以“；”号结束。

（4）<?php?>标识符的作用，就是告诉 Web 服务器，PHP 代码从什么地方开始，到什么地方结束。<?php ?>标识符内的所有文本都要按照 PHP 语言进行解释，以区别于 HTML 代码。

2.6　高手私房菜

技巧 1：如何设置网站的主目录？

在 Windows 7 操作系统中，设置网站的主目录的方法如下。

利用本章的方法打开【计算机管理】窗口，选择【Default Web Site】选项，如图 2-22 所示。

在右侧的窗格中单击【基本设置】链接，打开【编辑网站】对话框，单击【物理路径】下的 ... 按钮，即可在打开的对话框中重新设置网站的主目录，如图 2-23 所示。



图 2-22　【计算机管理】窗口

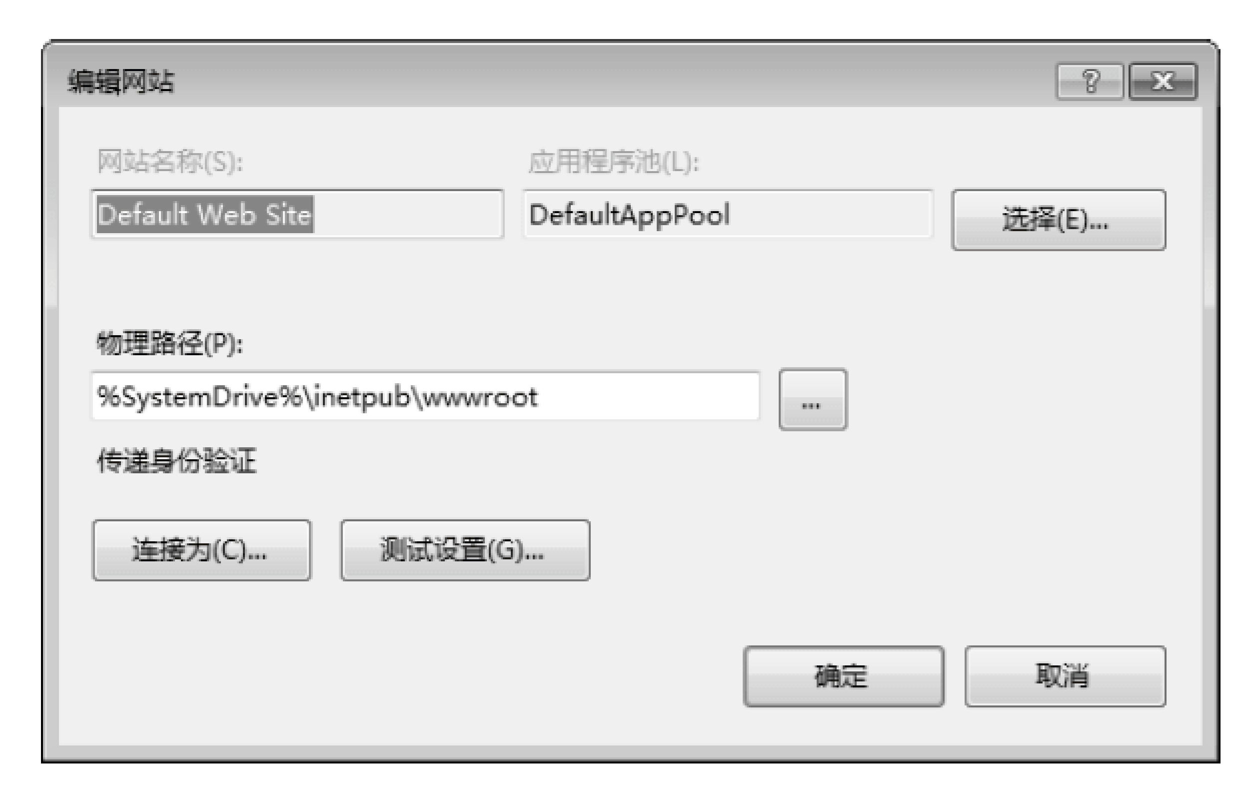


图 2-23　【编辑网站】对话框

技巧 2：启动 Apache 2.4 出错，提示缺少 msvcr110.dll 怎么办？

安装 Apache 2.4 之前，请用户务必安装 Visual C++ 2015，同时启动以下三个服务：windows modules installer、windows update 和 window defender service，否则启动 Apache 2.4 一定会失败。

2.7　经典习题

（1）配置 PHP 7+Apache 服务器环境。

（2）安装 WampServe 配置 PHP 7 服务器集成环境。

（3）测试 PHP 服务器是否配置成功。

第 3 章　PHP 7 的基本语法

上一章讲述了 PHP 7 的环境搭建方法，本章将开始学习 PHP 的基本语法，主要包括 PHP 的标识符、编码规范、常量、变量、数据类型、运算符和表达式等。通过本章的学习，读者可以掌握 PHP 的基本语法知识和技能。

本章学习目标

- 了解 PHP 标识符
- 熟悉 PHP 的编码规范
- 掌握常量的使用方法
- 掌握变量的使用方法
- 掌握数据类型
- 掌握运算符的使用方法
- 掌握表达式的使用方法
- 掌握创建多维数组的方法

3.1　PHP 标识符

默认情况下，PHP 是以<?php ?>标识符为开始和结束标记的。但是，PHP 代码有不同的表示风格。最为常用的风格就是这种默认方法，也有人把这种默认风格称为 PHP 的 XML 风格。下面将学习其他类型的标识风格。

3.1.1　短风格

有时候，读者会看到一些代码中出现用<? ?>标识符表示 PHP 代码的情况。这就是所谓的“短风格”（short style）表示法。例如：

```
<? echo "这是 PHP 短风格的表示方式。"?>
```

这种表示方法在正常情况下并不推荐，并且在 php.ini 文件中 short_open_tags 的默认设置是关闭的。另外，以后提到的一些功能设置会与这种表示方法相冲突，比如与 XML 的默认标识符相冲突。

3.1.2　script 风格

有的编辑器由于跟以前程序的定义表示要区分开，对 PHP 代码完全采用另外一种表示方式，即<script></script>的表示方式。例如：

```
<script language="php">
```

```
    echo "这是 PHP 的 script 表示方式。";
</script>
```

这十分类似于 HTML 页面中 JavaScript 的表示方式。

3.1.3 ASP 风格

由于 ASP 的影响，为了照顾 ASP 使用者对 PHP 的使用，PHP 提供了 ASP 的表示风格，例如：

```
<%
    echo "这是 PHP 的 ASP 的表示方式。";
%>
```

这种表示是在特殊情况下使用的，并不推荐正常使用。

3.2 编码规范

由于现在的 Web 开发往往是多人一起合作完成的，所以使用相同的编码规范显得非常重要，特别是新的开发人员参与时，通常需要知道前面开发代码中变量或函数的作用等，这就需要统一的编码规范。

3.2.1 什么是编码规范

编码规范是一套某种编程语言的导引手册，这种导引手册规定了一系列这种语言的默认编程风格，以用来增强这种语言的可读性、规范性和可维护性。一个语言的编码规范主要包括：此语言下的文件组织、缩进、注释、声明、空格处理、命名规则等。

遵守编码规范有以下好处。

- 编码规范是团队开发中对每个成员的基本要求。编码规范的好坏是一个程序员成熟程度的表现。
- 提高程序的可读性，有利于开发人员互相交流。
- 良好一致的编程风格，在团队开发中可以达到事半功倍的效果。
- 有助于程序的维护，降低软件成本。

3.2.2 PHP 中的编码规范

PHP 作为一种高级语言，十分强调编码规范。以下是此规范在 3 个方面的体现。

1. 表述

在 PHP 的正常表述中，每一句 PHP 语句都是以“；”结尾，这个规范就告诉 PHP 要执行此语句，例如：

```
<?php
   echo "php 以分号表示语句的结束和执行。";
?>
```

2. 指令分隔符

在 PHP 代码中，每个语句后需要用分号结束命令。一段 PHP 代码中的结束标记隐含表示了一个分号，所以在 PHP 代码段中的最有一行可以不用分号结束。例如：

```
<?php
   echo "这是第一个语句";        // 每个语句都加入分号
   echo "这是第二个语句";
  echo "这是最后一个语句"?>    // 结束标记“?>”隐含了分号，这里可以省略分号
```

3. 空白符

PHP 对空格、回车造成的新行、Tab 等留下的空白的处理也遵循编码规范。PHP 对它们都进行了忽略。这跟浏览器对 HTML 语言中的空白的处理是一样的。

合理利用空白符可以增强代码的可读性和清晰性。

（1）下列情况应该总是使用两个空白行：

① 两个类的声明之间。
② 一个源文件的两个代码片段之间。

（2）下列情况应该总是使用一个空白行：

① 两个函数声明之前。
② 函数内的局部变量和函数的第一个语句之间。
③ 块注释或单行注释之前。
④ 一个函数内的两个逻辑代码段之间。

（3）合理利用空格可以提高代码的缩进，提高可读性。

① 空格通常使用于关键字与括号之间，但是，函数名称与左括号之间不使用空格分开。
② 函数参数列表中的逗号后面通常会插入空格。
③ for 语句的表达式应该用逗号分开，后面添加空格。

4. 注释

为了增强可读性，在很多情况下，程序员都需要在程序语句的后面添加文字说明。而 PHP 要把它们与程序语句区分开，就需要让这些文字注释符合编码规范。

这些注释的风格有几种，分别是 C 语言风格、C++风格和 SHELL 风格。

C 语言风格如下：

```
/*这是 C 语言风格的注释内容*/
```

这种方法还可以多行使用：

```
/*这是
  C 语言风格
  的注释内容
*/
```

C++风格如下：

```
//这是 C++风格的注释内容行一
//这是 C++风格的注释内容行二
```

这种方法只能一句注释占用一行。使用时可单独一行，也可以使用在 PHP 语句之后的同一行。

SHELL 风格如下：

```
#这是 SHELL 风格的注释内容
```

这种方法只能一句注释占用一行。使用时可单独一行，也可以使用在 PHP 语句之后的同一行。

5. 与 HTML 语言混合搭配

凡是在一对 PHP 开始和结束标记之外的内容都会被 PHP 解析器忽略，这使得 PHP 文件可以具备混合内容。可以使 PHP 嵌入到 HTML 文档中去。例如：

```
<HTML>
<HEAD>
    <TITLE>PHP 与 HTML 混合</TITLE>
</HEAD>
<BODY>
<?php
      echo "嵌入的 PHP 代码";
?>
</BODY>
<HTML>
```

3.3 常　量

在 PHP 中，常量是一旦声明就无法改变的值。本节来讲述如何声明和使用常量。

3.3.1 声明和使用常量

PHP 通过 define()命令来声明常量，格式如下：

```
define ("常量名", 常量值);
```

常量名是一个字符串，通常在 PHP 的编码规范的指导下使用大写英文字母表示，比如 CLASS_NAME 、MYAGE 等。

常量值可以是很多种 PHP 的数据类型，可以是数组，可以是对象，当然也可以是字符和数字。

常量就像变量一样存储数值，但是与变量不同的是，常量的值只能设定一次，并且无论在代码的任何位置，它都不能被改动。常量声明后具有全局性，在函数内外都可以访问。

【例 3.1】（实例文件：ch03\3.1.php）

```
<?php
  define("HUANY","欢迎学习 PHP 基本语法知识");  // 定义常量 HUANY

   echo HUANY;  // 输出常量值
```

```
?>
```

本程序运行结果如图 3-1 所示。



图 3-1　运行结果

【案例分析】：

- 用 define 函数声明一个常量。常量的全局性体现在可在函数内外进行访问。
- 常量只能存储布尔值、整型、浮点型和字符串数据。

3.3.2　内置常量

PHP 的内置常量是指 PHP 在系统建立之初就定义好的一些量。PHP 中预定义了很多系统内置常量，这些常量可以被随时调用。下面列出了一些常见的内置常量。

1. _FILE_

这个默认常量是文件的完整路径和文件名。若引用文件（include 或 require）则在引用文件内的该常量为引用文件名，而不是引用它的文件名。

2. _LINE_

这个默认常量是 PHP 程序行数。若引用文件（include 或 require）则在引用文件内的该常量为引用文件的行，而不是引用它的文件行。

3. PHP_VERSION

这个内置常量是 PHP 程序的版本，如 3.0.8-dev。

4. PHP_OS

这个内置常量是指执行 PHP 解析器的操作系统名称，如 Linux。

5. TRUE

这个常量就是真值（true）。

6. FALSE

这个常量就是伪值（false）。

7. E_ERROR

这个常量指到最近的错误处。

8. E_WARNING

这个常量指到最近的警告处。

9. E_PARSE

本常量指到解析语法有潜在问题处。

10. E_NOTICE

这个常量为发生不寻常但不一定是错误处，例如存取一个不存在的变量。

11. _DIR_

这个常量为文件所在的目录。该常量在 PHP 5.3.0 版本中新增。

12. _ FUNCTION _

这个常量为函数的名称。从 PHP 5 开始，此常量返回该函数被定义时的名字，并且区分大小写。

13. _ CLASS _

这个常量为类的名称。从 PHP 5 开始，此常量返回该类被定义时的名字，并且区分大小写。

下面举例说明系统常量的使用方法。

【例 3.2】（实例文件：ch03\3.2.php）

```
<?php
 echo(_FILE_);          // 输出文件的路径和文件名
 echo “<br />“;           // 输出换行
 echo(_LINE_);          // 输出语句所在的行数
 echo “<br />“;
 echo(PHP_VERSION); // 输出 PHP 的版本
 echo “<br />“;
 echo(PHP_OS);       // 输出操作系统名称
?>
```

本程序运行结果如图 3-2 所示。

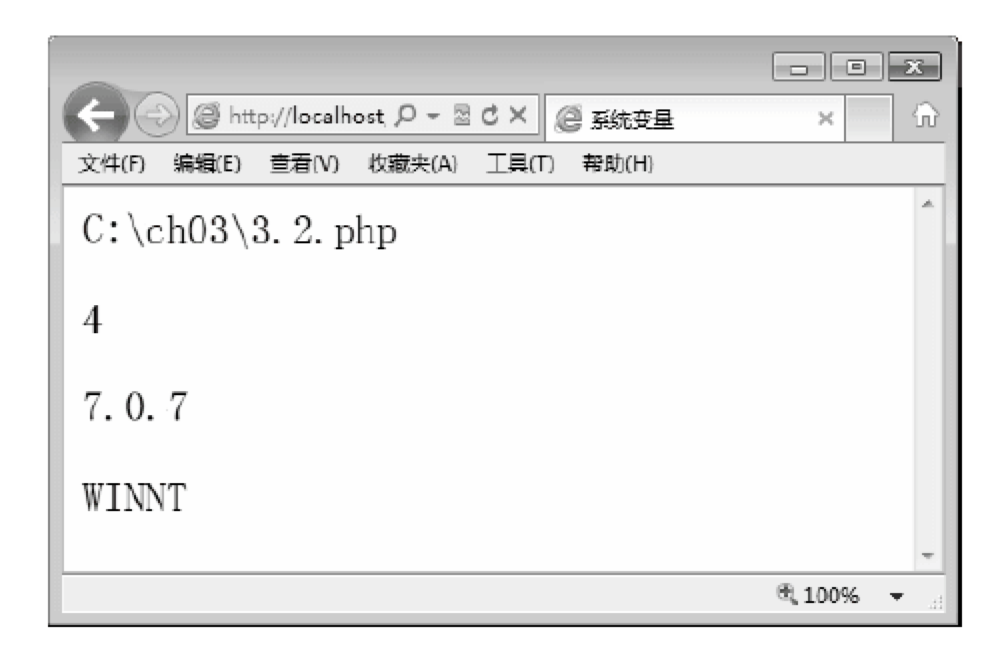


图 3-2　程序运行结果

【案例分析】：

（1）echo “
“语句表示为输出换行。

（2）echo(_FILE_)语句输出文件的文件名，包括详细的文件路径。echo(_LINE_)语句输出该语句所在的行数。echo(PHP_VERSION)语句输出 PHP 程序的版本。echo(PHP_OS)语句输出执行 PHP 解析器的操作系统名称。

3.4 变　量

变量像一个贴有名字标签的空盒子。不同的变量类型对应不同种类的数据，就像不同种类的东西要放入不同种类的盒子。

3.4.1 PHP 中的变量声明

PHP 中的变量不同于 C 或 Java 语言，因为它是弱类型的。在 C 或 Java 中，需要对每一个变量声明类型，但是在 PHP 中不需要这样做。

PHP 中的变量一般以“$”作为前缀，然后以字母 a~z 的大小写或者“_”下划线开头。这是变量的一般表示。

合法的变量名可以是：

```
$hello
$Aform1
$_formhandler （类似我们见过的$_POST 等）。
```

非法的变量名如：

```
$168
$!like
```

PHP 中不需要显式地声明变量，但是定义变量前进行声明并带有注释，这是一个好的程序员应该养成的习惯。PHP 的赋值有两种，包括传值和引用，它们的区别如下：

（1）传值赋值：使用“=”直接将赋值表达式的值赋给另一个变量。

（2）引用赋值：将赋值表达式内存的空间的引用赋给另一个变量。需要在“=”左右的变量前面加上一个“&”符号。在使用引用赋值的时候，两个变量将会指向内存中同一个存储空间，所以任意一个变量的变化都会引起另外一个变量的变化。

【例 3.3】（实例文件：ch03\3.3.php）

```
<?php
echo "使用传值方式赋值：<br/>";        // 输出 使用传值方式赋值：
$a = "风吹草低见牛羊";
$b = $a;                    // 将变量$a 的值赋值给$b，两个变量指向不同内存空间
echo "变量 a 的值为".$a."<br/>";        // 输出 变量 a 的值
echo "变量 b 的值为".$b."<br/>";        // 输出 变量 b 的值
$a = "天似穹庐，笼盖四野";      // 改变变量 a 的值，变量 b 的值不受影响
echo "变量 a 的值为".$a."<br/>";        // 输出 变量 a 的值
echo "变量 b 的值为".$b."<p>";          //输出 变量 b 的值
echo "使用引用方式赋值：<br/>";        //输出  使用引用方式赋值：
$a = "天苍苍，野茫茫";
$b = &$a;                   // 将变量$a 的引用赋给$b，两个变量指向同一块内存空间
```

```
    echo "变量 a 的值为".$a."<br/>";     // 输出 变量 a 的值
    echo "变量 b 的值为".$b."<br/>";     // 输出 变量 b 的值
    $a = "敕勒川，阴山下";
    /*
    改变变量 a 在内存空间中存储的内容，变量 b 也指向该空间，b 的值也发生变化
    */
    echo "变量 a 的值为".$a."<br/>";   // 输出 变量 a 的值
    echo "变量 b 的值为".$b."<p>";     // 输出 变量 b 的值
    ?>
```

本程序运行结果如图 3-3 所示。

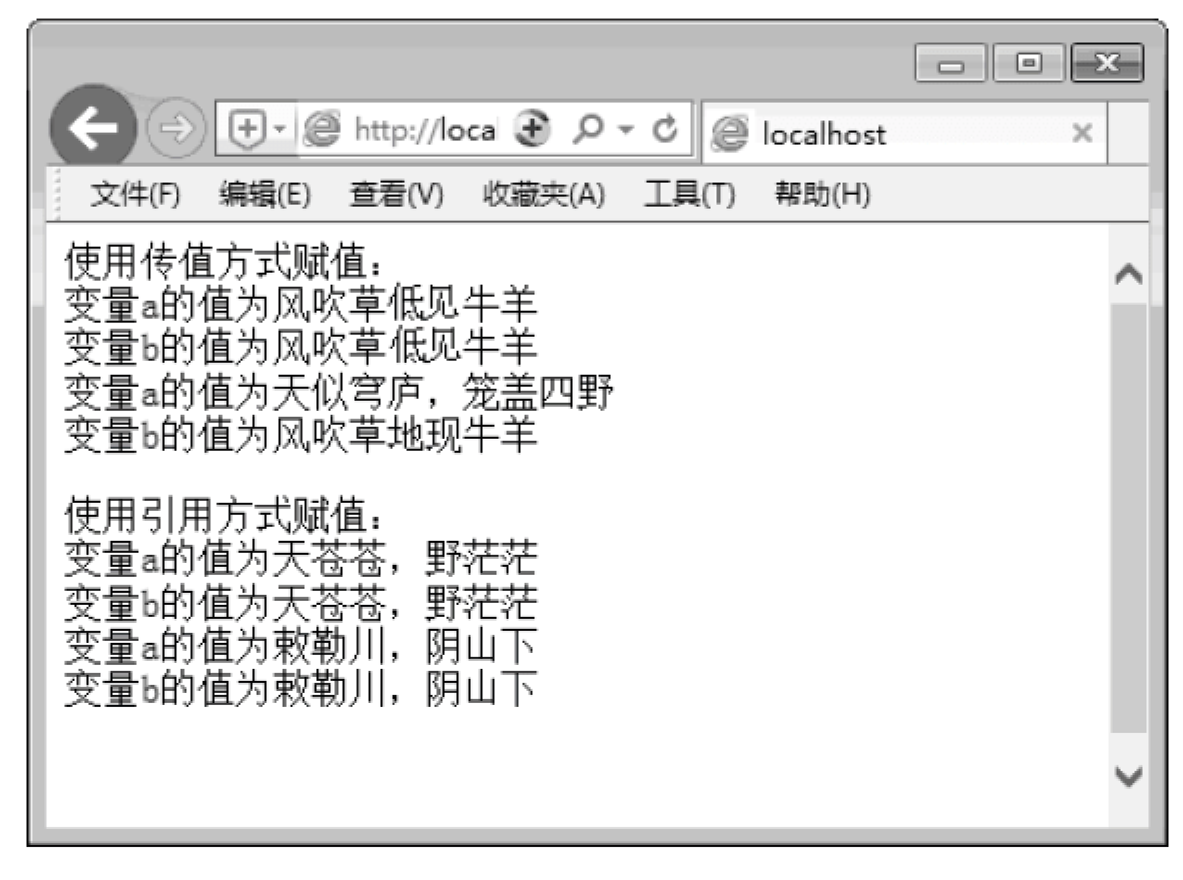


图 3-3　程序运行结果

3.4.2　可变变量与变量的引用

一般的变量表示很容易理解，但是有两个变量表示概念比较容易混淆，就是可变变量和变量的引用。通过以下例子对它们进行说明。

【例 3.4】（实例文件：ch03\3.4.php）

```
<?php
  $value0 = "guest";                 // 定义变量$value0 并赋值
  $$value0 = "customer";             // 再次给变量赋值
  echo $guest."<br />";              // 输出变量
  $guest = "feifei";                 // 定义变量$guest 并赋值
  echo $guest."\t".$$value0."<br />";
  $value1 = "xiaoming";              // 定义变量$value1
  $value2 = &$value1;               // 引用变量并传递变量
  echo $value1."\t".$value2."<br />";  // 输出变量
  $value2 = "lili";
  echo $value1."\t".$value2;
?>
```

本程序运行结果如图 3-4 所示。

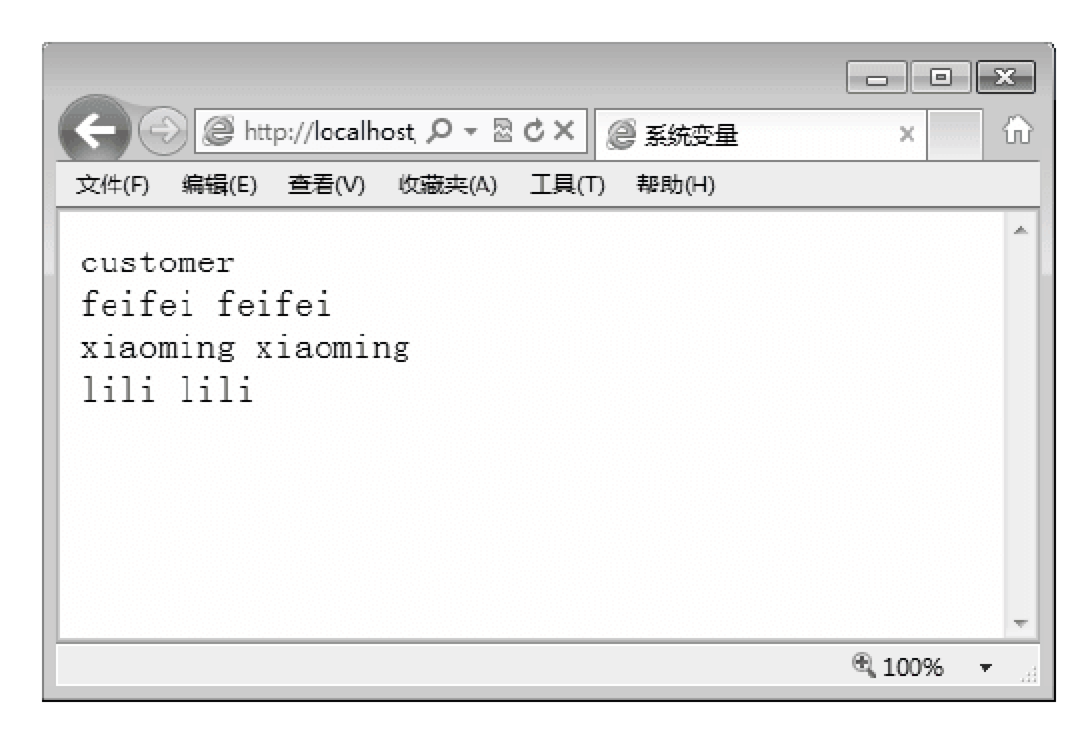


图 3-4　程序运行结果

【案例分析】：

（1）在代码的第一部分，$value0 被赋值 guest。$value0 相当于 guest，则$$value0 相当于$guest。所以当$$value0 被复制为 customer 时，打印$guest 就得到 customer。反之，当$guest 变量被赋值为 feifei 时，打印$$value0 同样得到 feifei。这就是可变变量。

（2）在代码的第二部分里，$value1 被赋值 xiaoming，然后通过“&”引用变量$value1 并赋值给$value2。而这一步的实质是，给变量$value1 添加了一个别名$value2。所以打印时，大家都得出原始赋值 xiaoming。由于$value2 是别名，和$value1 指的是同一个变量，所以当$value2 被赋值 lili 后，$value1 和$value2 都得到新值 lili。

（3）可变变量，其实是允许改变一个变量的变量名。允许使用一个变量的值作为另外一个变量的名。

（4）变量引用，相当于给变量添加了一个别名，使用“&”来引用变量。其实两个变量名指的都是同一个变量。就像是给同一个盒子贴了两个名字标签，两个名字标签指的都是同一个盒子。

3.4.3　变量作用域（variable scope）

所谓变量作用域（scope），是指特定变量在代码中可以被访问到的位置。在 PHP 中有 6 种基本的变量作用域法则。

（1）内置超全局变量（Built-in superglobal variables），在代码中的任意位置都可以访问到。

（2）常数（constants），一旦声明，它就是全局性的。可以在函数内外使用。

（3）全局变量（global variables），在代码间声明，可在代码间访问，但是不能在函数内访问。

（4）在函数中声明为全局变量的变量，就是同名的全局变量。

（5）在函数中创建和声明为静态变量的变量，在函数外是无法访问的。但是这个静态变量的值可以保留。

（6）在函数中创建和声明的局部变量，在函数外是无法访问的，并且在本函数终止时失效。

1. 超全局变量

Superglobal 或 autoglobal 可以称为“超全局变量”或“自动全局变量”。这种变量的特性是，不管在程序的任何地方都可以访问到，不管是函数内或是函数外也都可以访问到。而这些“超全局变量”就是由 PHP 预先定义好以方便使用的。

那么这些“超全局变量”或“自动全局变量”都有哪些呢？

- $GLOBALS：包含全局变量的数组。
- $_GET：包含所有通过 GET 方法传递给代码的变量的数组。
- $_POST：包含所有通过 POST 方法传递给代码的变量的数组。
- $_FILES：包含文件上传变量的数组。
- $_COOKIE：包含 cookie 变量的数组。
- $_SERVER：包含服务器环境变量的数组。
- $_ENV：包含环境变量的数组。
- $_REQUEST：包含用户所有输入内容的数组（包括$_GET、$_POST 和$_COOKIE）。
- $_SESSION：包含会话变量的数组。

2. 全局变量

全局变量其实就是在函数外声明的变量，在代码间都可以访问，但是在函数内是不能访问的。这是因为函数默认不能访问在其外部的全局变量。

通过下面的实例介绍全局变量的使用方法和技巧。

【例 3.5】（实例文件：ch03\3.5.php）

```
<?php
  $room = 20;                  // 定义全局变量
  function showrooms(){
      echo $room;              // 函数内访问全局变量
  }
  showrooms();                // 访问全局变量
  echo $room.'间房间。';
?>
```

本程序运行结果如图 3-5 所示。

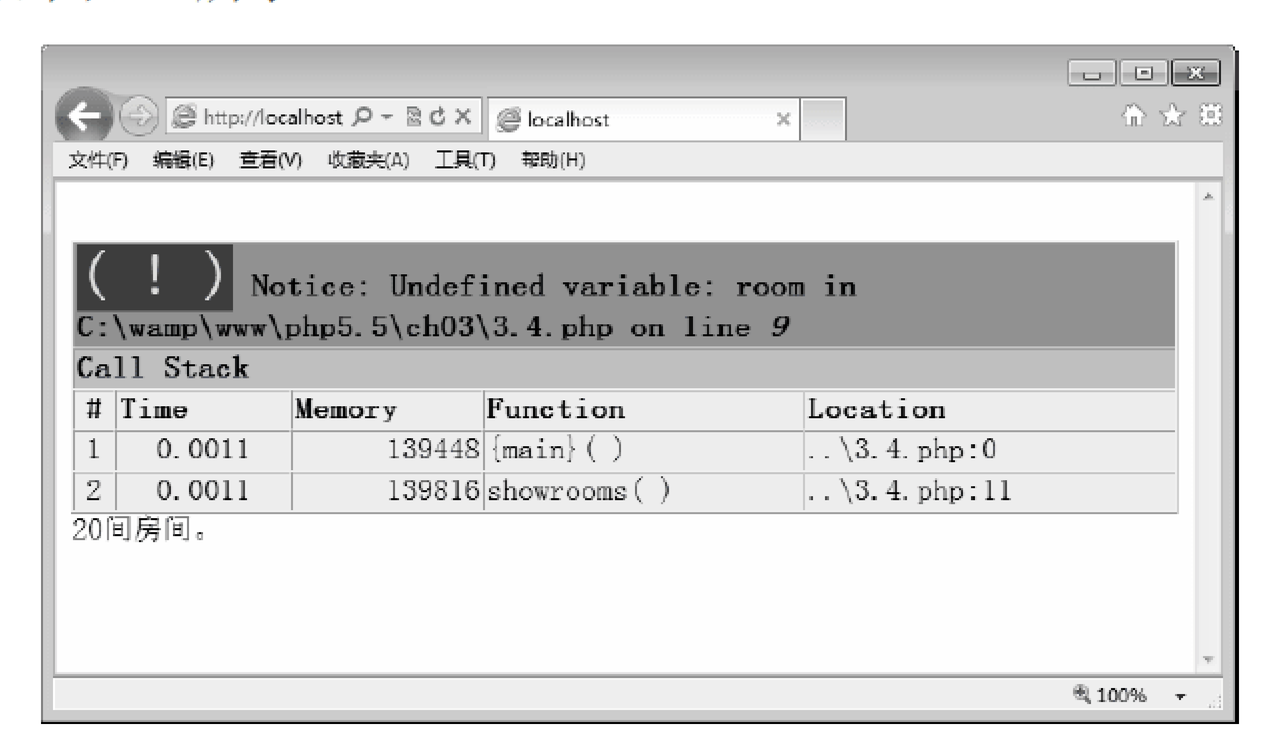


图 3-5　程序运行结果

出现上述结果，是因为函数无法访问外部全局变量，但是在代码间可以访问全局变量。

如果想让函数访问某个全局变量，可以在函数中通过 global 关键字来声明，就是要告诉函数，它要调用的变量是一个已经存在或者即将创建的同名全局变量，而不是默认的本地变量。

通过下面的实例介绍 global 关键字的使用方法和技巧。

【例 3.6】（实例文件：ch03\3.6.php）

```
<?php
  $room = 20;                          // 定义全局变量
  function showrooms(){
      global $room;                     //函数内调用全局变量
      echo $room.'间新房间。<br />';
  }
  showrooms();
  echo $room.'间房间。';
?>
```

本程序运行结果如图 3-6 所示。

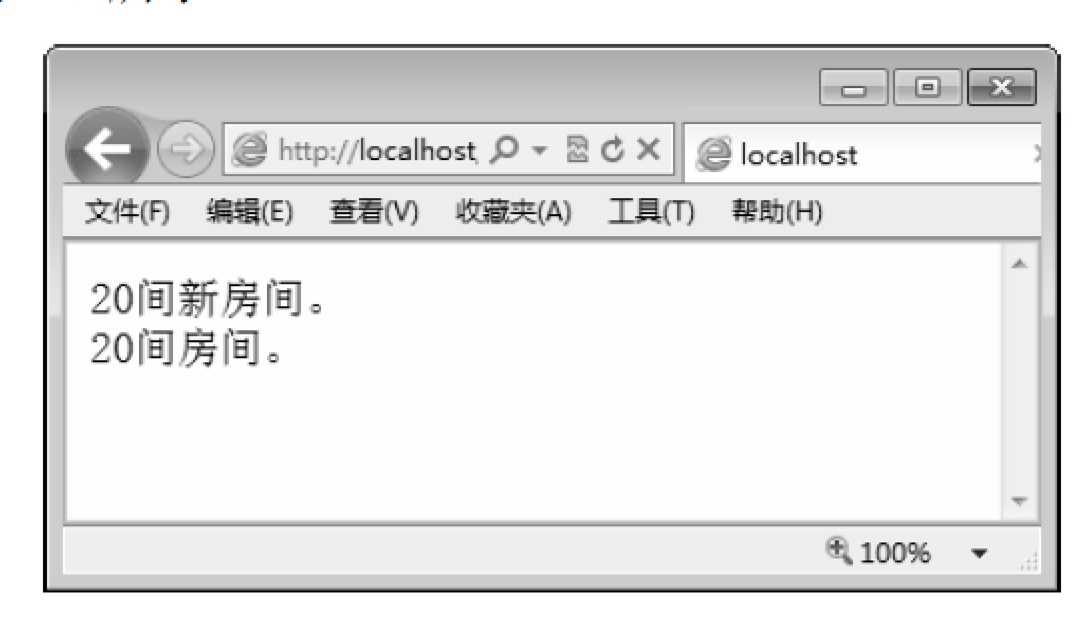


图 3-6　程序运行结果

另外，读者还可以通过“超全局变量”中的$GLOBALS 数组进行访问。

下面通过以下实例介绍$GLOBALS 数组，具体步骤如下。

【例 3.7】（实例文件：ch03\3.7.php）

```
<?php
  $room = 20;                          // 定义全局变量
  function showrooms(){
      $room = $GLOBALS['room'];       // 通过$GLOBALS 数组进行访问全局变量
      echo $room.'间新房间。<br />';
  }
  showrooms();
  echo $room.'间房间。';
?>
```

本程序运行结果如图 3-7 所示。

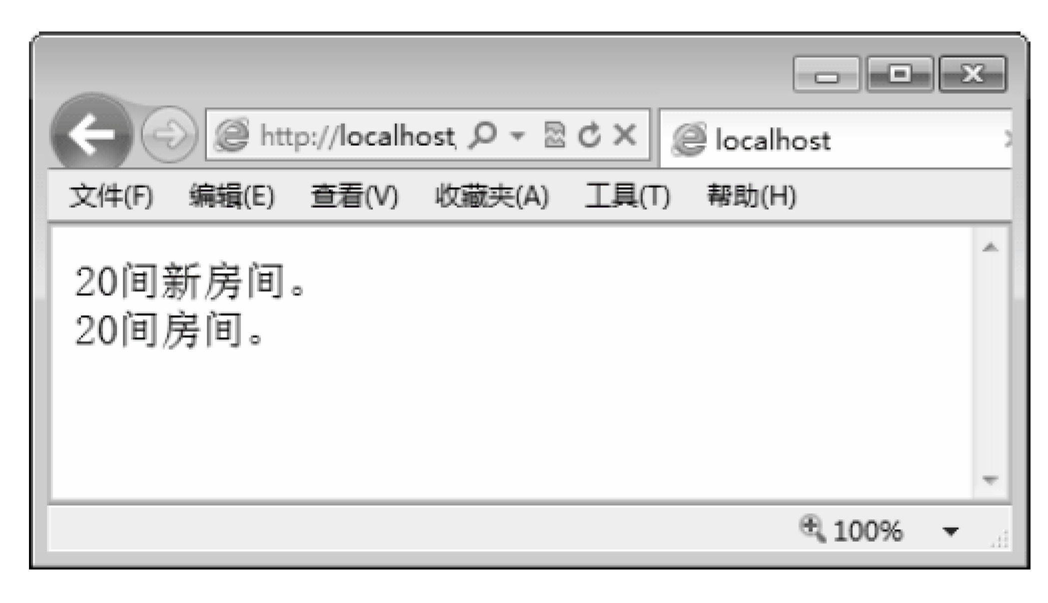


图 3-7　程序运行结果

结果与上面的例子完全相同，可见这两种方法可以实现同一个效果。

3. 静态变量

静态变量只是在函数内存在，在函数外无法访问。但是执行后，其值保留，也就是说这一次执行完毕后，静态变量的值保留，下一次再执行此函数，这个值还可以调用。

通过下面的实例介绍静态变量的使用方法和技巧。

【例 3.8】（实例文件：ch03\3.8.php）

```
<?php
   $person = 20;                       // 输出操作系统名称
  function showpeople(){
      static $person = 5;
    $person++;
      echo '再增加一位，将会有 '.$person.' 位 static 人员。<br />';
  }
  showpeople();
  echo $person.' 人员。<br />';
  showpeople();
?>
```

本程序运行结果如图 3-8 所示。

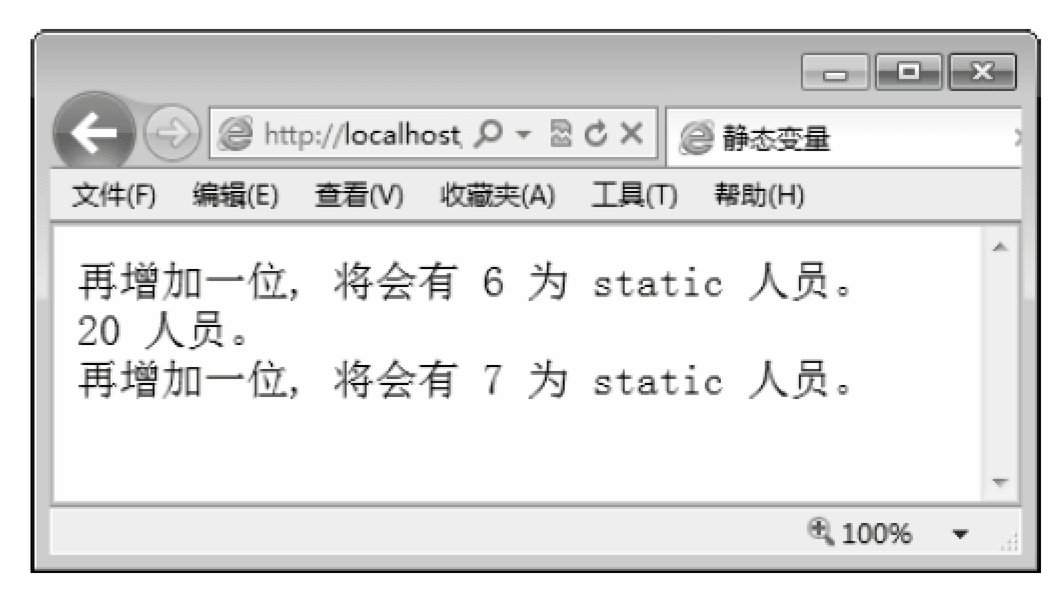


图 3-8　程序运行结果

【案例分析】：

（1）其中函数外的 echo 语句无法调用函数内的 static $person，它调用的是$person = 20。

（2）showpeople()函数被执行两次，这个过程中 static $person 的运算值得以保留，并且通过$person++进行了累加。

3.4.4　变量的销毁

当用户创建一个变量时，相应的在内存中有一个空间专门用于存储该变量，该空间引用计数加 1。当变量与该空间的联系被断开时，则空间引用计数减 1，直到引用计数为 0，则称为垃圾。

PHP 有自动回收垃圾的机制，用户也可以手动销毁变量，通常使用 unset()函数来实现。该函数的语法格式如下：

```
void unset （变量）
```

其中变量类型为局部变量，则变量被销毁；如果变量类型为全局变量，则变量不会被销毁。

【例 3.9】（实例文件：ch03\3.9.php）

```
<?php
function xiaohui()  {     //声明函数
    global $b;     //函数内使用 global 关键字声明全局变量$b
    unset ($b);       //使用 unset()销毁不再使用的变量$b
}
$b= '春江花月夜';                 //函数外声明全局变量
xiaohui();            //调用函数
echo  $b;            //查看全局变量是否发生变化
?>
```

本程序运行结果如图 3-9 所示。



图 3-9　程序运行结果

3.5　数据的类型

从 PHP 4 开始，PHP 中的变量不需要事先声明，赋值即可声明。在使用这些数据类型前，读者需要了解它们的含义和特性。下面介绍整型、浮点型、布尔型、字符串型、数组型、对象型和两个较特殊的类型。

3.5.1　什么是类型

不同的数据类型其实就是所存储数据的不同种类。PHP 的不同数据类型主要包括：

- 整型（integer），用来存储整数。
- 浮点型（float），用来存储实数。
- 字符串型（string），用来存储字符串。
- 布尔型（boolean），用来存储真（true）或假（false）。
- 数组型（array），用来存储一组数据。
- 对象型（object）， 用来存储一个类的实例。

作为弱类型语言，PHP 也被称为动态类型语言。在强类型语言，例如 C 语言中，一个变量只能存储一种类型的数据，并且这个变量在使用前必须声明变量类型。而在 PHP 中，给变量赋什么类型的值，这个变量就是什么类型，例如以下几个变量：

```
$hello = "hello world";
```

由于 hello world 是字符串，则变量$hello 的数据类型就是字符串类型。

```
$hello = 100;
```

同样，由于 100 为整型，$hello 也就是整型。

```
$wholeprice = 100.0;
```

由于 100.0 为浮点型，则$wholeprice 就是浮点型。

由此可见，对于变量而言，如果没有定义变量的类型，则它的类型由所赋值的类型决定。

3.5.2 整型（integer）

整型（integer）是数据类型中最为基本的类型。在现有的 32 位运算器的情况下，整型的取值是从-2147483648 到+2147483647。整型可以表示为十进制、十六进制和八进制。

例如：

```
3560     //十进制整数
01223    //八进制整数
0x1223   //十六进制整数
```

3.5.3 浮点型（float 或 double）

浮点型（floating-point）就是表示实数。在大多数运行平台下，这个数据类型的大小为 8 个字节。它的近似取值范围是 2.2E-308~1.8E+308（科学计数法）。

例如：

```
-1.432
1E+07
0.0
```

3.5.4 布尔型（boolean）

布尔型（boolean）只有两个值，就是 true 和 false。布尔型是十分有用的数据类型，通过它，程序实现了逻辑判断的功能。

其他的数据类型基本都有布尔属性：

- 整型，为 0 时，其布尔属性为 false，为非零值时，其布尔属性为 true。
- 浮点型，为 0.0 时，其布尔属性为 false，为非零值时，其布尔属性为 true。
- 字符串型，为空字符串“”，或者零字符串“0”时，其布尔属性为 false，包含除此以外的字符串时其布尔属性为 true。
- 数组型，若不含任何元素，其布尔属性为 false，只要包含元素，则其布尔属性为 true。
- 对象型，资源型，其布尔属性永远为 true。
- 空型，其布尔属性永远为 false。

3.5.5 字符串型（string）

字符串型的数据是表示在引号之间的。引号分为双引号（" "）和单引号（' '）。这两种引号可以表示字符串。但是这两种表示方法也有一定区别。

双引号几乎可以包含所有的字符，但是在其中的变量显示变量的值，而不是变量的变量名，而有些特殊字符加上“\”符号就可以了；单引号内的字符是被直接表示出来的。

下面通过一个案例，来讲述整型、浮点型、布尔型和字符串型数据的使用方法和技巧。

【例 3.10】（实例文件：ch03\3.10.php）

```
<?php
   $int1= 2012;      // 十进制整数
   $int2= 01223;     //八进制整数
   $int3=0x1223;    //十六进制整数
   echo "输出整数类型的值：";
   echo $int1;
   echo "\t";    //输出一个制表符
   echo $int2;    //输出 659
   echo "\t";
   echo $int3;    //输出 4643
   echo "<br />";
   $float1=54.66;
   echo $float1;    //输出 54.66
   echo "<br />";
   echo "输出布尔型变量：";
   echo (Boolean)( $int1);     //将 int1 整型转化为布尔变量
   echo "<br />";
   $string1="字符串类型的变量";
   echo $string1;
?>
```

本程序运行结果如图 3-10 所示。

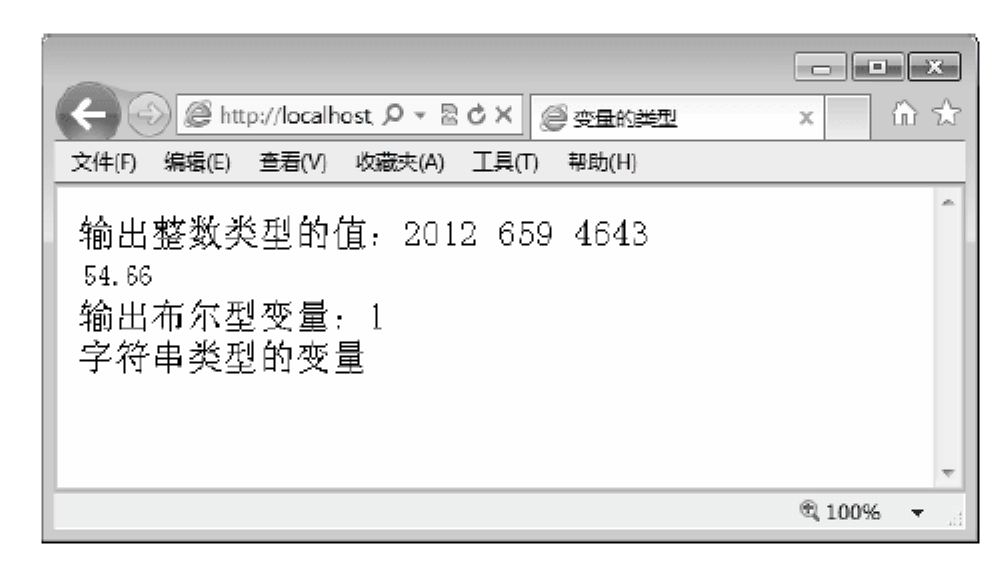


图 3-10　程序运行结果

3.5.6　数组型（array）

数组是 PHP 变量的集合，它是按照“键值”与“值”的对应关系组织数据的。数组的键值可以是整数，也可以是字符串。另外，数组不特意表明键值的默认情况下，数组元素的键值为从零开始的整数。

在 PHP 中，使用 list()函数或 array()函数来创建数组，也可以直接进行赋值。

下面使用 array()函数创建数组。

【例 3.11】（实例文件：ch03\3.11.php）

```
<?php
```

```
$arr=array                              // 定义数组并赋值
 (
    0=>15,
    2=>1E+05,
    1=>"开始学习 PHP 基本语法了",
);
for ($i=0;$i<count($arr);$i++)    // 使用 for 循环输出数组内容
{
  $arr1=each($arr);
  echo "$arr1[value]<br />";
}
?>
```

本程序运行结果如图 3-11 所示。



图 3-11　程序运行结果

【案例分析】：

（1）程序中用“=>”为数组赋值，数组的下标只是存储的标识，没有任何意义，数组元素的排列以加入的先后顺序为准。

（2）本程序采用 for 循环语句输出整个数组，其中 count 函数返回数组的个数，echo 函数返回当前数组指针的索引值对，后面的章节将详细讲述 echo 函数的使用方法。

上面实例的语句可以简化如下。

【例 3.12】（实例文件：ch03\3.12.php）

```
<?php
  $arr=array(15,1E+05,"开始学习 PHP 基本语法了");    // 定义数组并赋值
   for ($i=0;$i<3;$i++)
      {
       echo $arr[$i]<br />";
       }
?>
```

本程序运行结果如图 3-12 所示。从结果可以看出，这两种写法的运行结果相同。



图 3-12　程序运行结果

另外，读者还可以对数组的元素一个一个地赋值，下面举例说明。上面的语句可以简化如下。

【例 3.13】（实例文件：ch03\3.13.php）

```
<?php
  $arr[0]=15;  // 对数组元素分别赋值
  $arr[2]= 1E+05;
  $arr[1]= "开始学习 PHP 基本语法了";
for ($i=0;$i<count($arr);$i++)
{
  $arr1=each($arr);
  echo "$arr1[value]<br />";
}
?>
```

本程序运行结果如图 3-13 所示。从结果可以看出，一个一个赋值的方法和上面两种写法的运行结果一样。



图 3-13　程序运行结果

3.5.7　对象型（object）

对象就是类的实例。当一个类被实例化以后，这个被生成的对象被传递给一个变量，这个变量就是对象型变量。对象型变量也属于资源型变量。

3.5.8　NULL 型

NULL 类型是仅拥有 NULL 这个值的类型。这个类型用来标记一个变量为空。一个空字符串与 NULL 是不同的。在数据库存储时会把空字符串和 NULL 区分开处理。NULL 型在布尔判断时永远为 false。很多情况下，在声明一个变量的时候可以直接先赋值为 NULL 型，如$value = NULL。

3.5.9　资源类型（resource）

Resource 类型，也就是资源类型，是十分特殊的数据类型。它表示 PHP 的扩展资源，可以是一个打开的文件，也可以是一个数据库连接，甚至可以是其他的数据类型。但是在编程过程中，资源类型却是几乎永远接触不到的。

3.5.10　数据类型之间的相互转换

数据从一种类型转换到另外一种类型，就是数据类型转换。在 PHP 语言中，有两种常见的转换方式：自动数据类型转换和强制数据类型转换。

1. 自动数据类型转换

这种转换方法最为常用，直接输入数据的转换类型即可。例如 float 型转换为整数 int 型，小数点后面的数将被舍弃。如果 float 数值超过了整数的取值范围，则结果可能是 0 或者整数的最小负数。

【例 3.14】（实例文件：ch03\3.14.php）

```
<?php
  $flo1=1.86;            // 定义 float 类型
  echo (int)$flo1."<br />";  // 转换为整数类型输出
  $flo2=4E32;          // 超过整数取值范围
  echo(int)$flo2;
?>
```

本程序运行结果如图 3-14 所示。

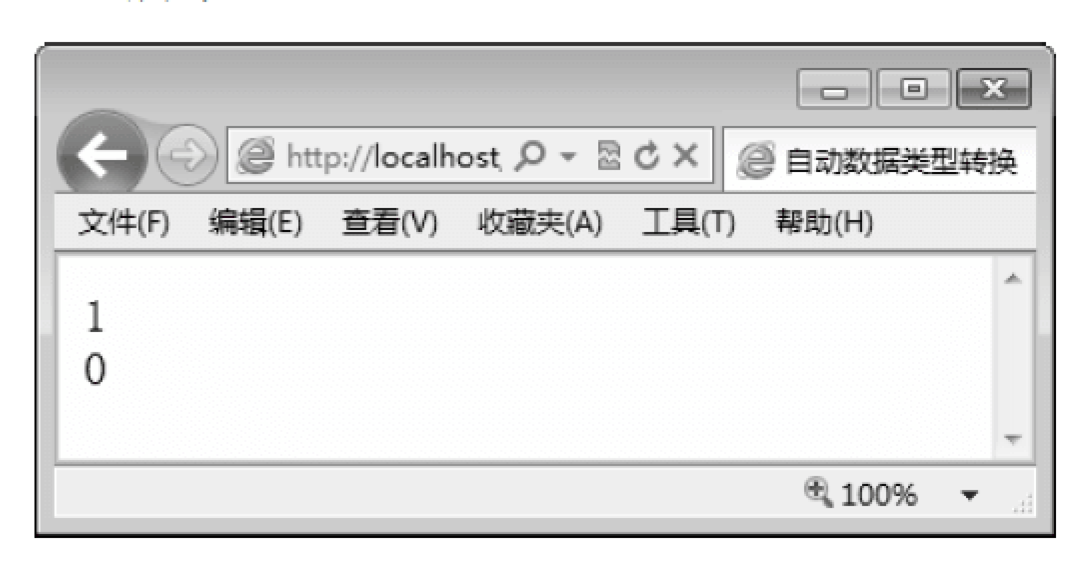


图 3-14　程序运行结果

2. 强制数据类型转换

在 PHP 中，可以使用 settype 函数强制转换数据类型，基本语法如下。

```
Bool settype(var,string type)
```

注意：type 的可能值不能包含资源类型数据。

【例 3.15】（实例文件：ch03\3.15.php）

```
<?php
    $flo1=1.86;  // 定义浮点型数据
    echo settype($flo1,"int");  强制转换数据为整数并输出
?>
```

本程序运行结果如图 3-15 所示。

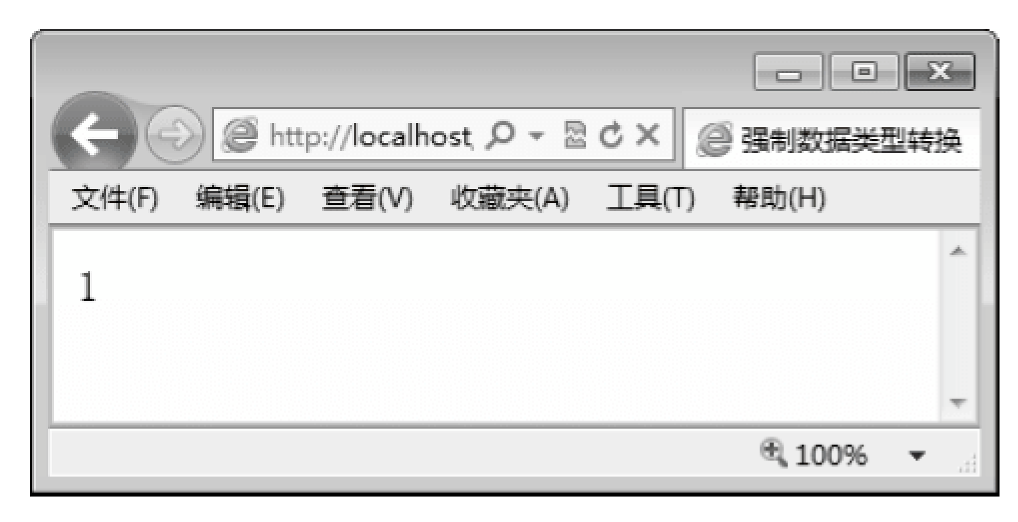


图 3-15　程序运行结果

3.6　标量类型的声明

默认情况下，所有的 PHP 文件都处于弱类型校验模式。PHP 7 加了标量类型声明的特性，标量类型声明有两种模式：强制模式（默认）和严格模式。

标量类型声明语法格式如下：

```
declare(strict_types=1);
```

通过指定 strict_types 的值（1 或者 0），1 表示严格类型校验模式，作用于函数调用和返回语句；0 表示弱类型校验模式。



提　示

可以声明标量类型的参数类型包括 int、float、bool、string、interfaces、array 和 callable。

1. 强制模式

下面通过案例来学习强制模式的含义，代码如下：

```
<?php
// 强制模式
function sum(int $ints)
{
   return array_sum($ints);
}
print(sum(2, '3', 4.1));
?>
```

上面程序输出结果为 9。代码中的 4.1 先转换为整数 4，然后再进行相加操作。

2. 严格模式

下面通过案例来学习严格模式的含义，代码如下：

```
<?php
// 严格模式
declare(strict_types=1);
function sum(int  $ints)
{
   return array_sum($ints);
}
print(sum(2, '3', 4.1));
?>
```

以上程序由于采用了严格模式，所以如果参数中出现不是整数的类型，程序执行时会报错。

3.7 运算符

PHP 包含多种类型的运算符，常见的有算术运算符、字符串运算符、赋值运算符、比较运算符和逻辑运算符等。

3.7.1 算术运算符

算术运算符是最简单，也是最常用的运算符。常见的算术运算符如表 3-1 所示。

表 3-1 常见的算术运算符

运算符	名称
+	加法运算符
-	减法运算符
*	乘法运算符
/	除法运算符
%	取余运算符
++	累加运算符
--	累减运算符

算术运算符的用法如下面的实例所示。

【例 3.16】（实例文件：ch03\3.16.php）

```
<?php
  $a=13;              // 定义变量
  $b=2;
  echo $a."+".$b."=";
  echo $a+$b."<br />";  //使用加法运算符
  echo $a."-".$b."=";
  echo $a-$b."<br />";  //使用减法运算符
  echo $a."*".$b."=";
  echo $a*$b."<br />";  //使用乘法运算符
  echo $a."/".$b."=";
```

```
echo $a/$b."<br />";  //使用除法运算符
echo $a."%".$b."=";
echo $a%$b."<br />";  //使用求余运算符
echo $a."++"."=";
echo $a++."<br />";  //使用累加运算符
echo $a."--"."=";
echo $a--."<br />";  //使用累减运算符
?>
```

本程序运行结果如图 3-16 所示。

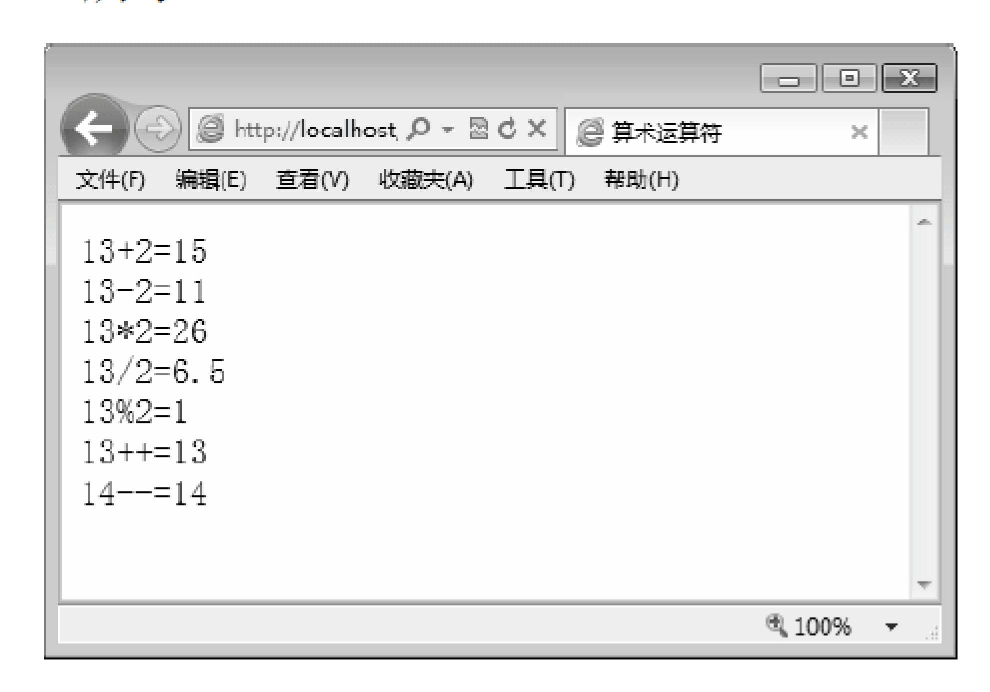


图 3-16　程序运行结果



提　示

除了数值可以进行自增运算外，字符也可以进行自增运算操作。例如 b++，结果将等于 c。

3.7.2　字符串运算符

字符串运算符是把两个字符串连接起来变成一个字符串的运算符，使用“.”来完成。如果变量是整型或浮点型，PHP 也会自动把它们转换为字符串输出，如下面的实例所示。

【例 3.17】（实例文件：ch03\3.17.php）

```
<?php
 $a = "把两个字符串";           // 定义字符串变量
 $b = 10.25;
  echo $a."连接起来，".$b."天。"; // 把字符串连接后输出
?>
```

本程序运行结果如图 3-17 所示。



图 3-17　程序运行结果

3.7.3 赋值运算符

赋值运算符的作用是把一定的数据值加载给特定变量。

赋值运算符的具体含义如表 3-2 所示。

表 3-2 赋值运算符的含义

赋值运算符	含义
=	将右边的值赋值给左边的变量
+=	将左边的值加上右边的值赋给左边的变量
-=	将左边的值减去右边的值赋给左边的变量
*=	将左边的值乘以右边的值赋给左边的变量
/=	将左边的值除以右边的值赋给左边的变量
.=	将左边的字符串连接到右边
%=	将左边的值对右边的值取余数赋给左边的变量

例如，$a-=$b 等价于$a=$a-$b，其他赋值运算符与之类似。从上表可以看出，赋值运算符可以使程序更加简练，从而提高执行效率。

3.7.4 比较运算符

比较运算符用来比较其两端数据值的大小。比较运算符的具体含义如表 3-3 所示。

表 3-3 比较运算符的含义

比较运算符	含义
==	相等
! =	不相等
>	大于
<	小于
>=	大于等于
<=	小于等于
===	精确等于（类型也要相同）
!==	不精确等于

其中，“===”和“!==”需要特别注意一下。$b===$c 表示$b 和$c 不只是数值上相等，而且两者的类型也一样；$b!==$c 表示$b 和$c 有可能是数值不等，也可能是类型不同。

【例 3.18】（实例文件：ch03\3.18.php）

```
<?PHP
$value="15";
echo "\$value = \"$value\"";
echo "\$value==15: ";
var_dump($value==15);        //结果为:bool(true)
echo "\$value==ture: ";
var_dump($value==false);        //结果为:bool(false)
echo "\$value!=null: ";
```

```
    var_dump($value!=null);           //结果为:bool(true)
    echo "\$value==false: ";
    var_dump($value==false);          //结果为:bool(false)
    echo "\$value === 100: ";
    var_dump($value===100);           //结果为:bool(false)
    echo "\$value===true: ";
    var_dump($value===true);          //结果为:bool(false)
    echo "(10/2.0 !== 5): ";
    var_dump(10/2.0 !==5);            //结果为:bool(true)
    ?>
```

本程序运行结果如图 3-18 所示。

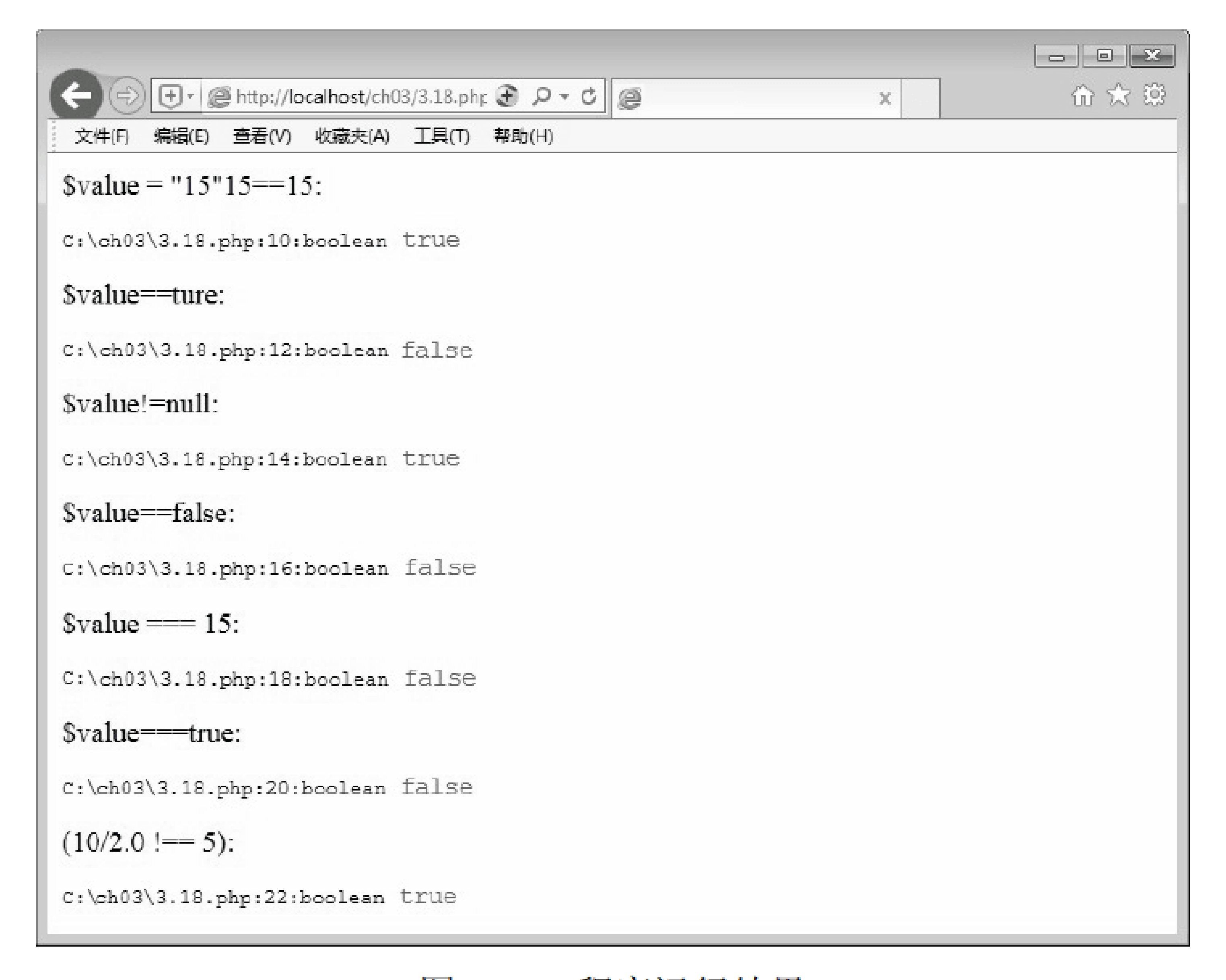


图 3-18　程序运行结果

3.7.5　逻辑运算符

编程语言最重要的功能之一就是进行逻辑判断和运算。逻辑和、逻辑或、逻辑否都是逻辑运算符。逻辑运算符的含义如表 3-4 所示。

表 3-4　逻辑运算的含义

逻辑运算符	含义
&&	逻辑和
AND	逻辑和
||	逻辑或
OR	逻辑或
!	逻辑否
NOT	逻辑否
XOR	逻辑异或

【例 3.19】（实例文件：ch03\3.19.php）

```
<?php
  $a = true;
  $b = false;
  echo '$a && $b: ';
  var_dump($a && $b);        //使用逻辑与运算符，结果为:false
  echo '$a || $b: ';
  var_dump($a || $b);        //使用逻辑或运算符，结果为:true
  echo '! $a: ';
  var_dump($a && $b);        //使用逻辑否运算符，结果为:false
?>
```

本程序运行结果如图 3-19 所示。

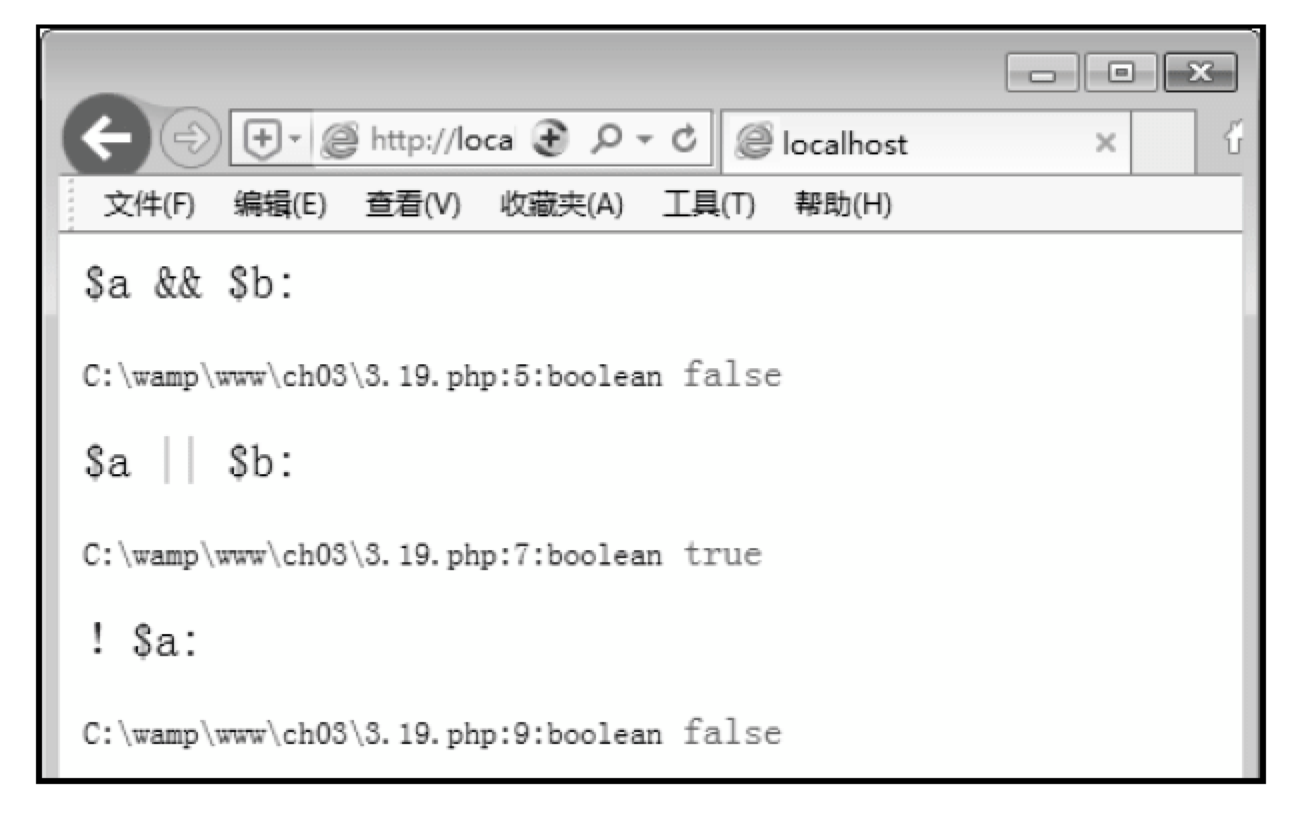


图 3-19　程序运行结果

3.7.6　按位运算符

按位运算符，是把整数按照“位”的单位来进行处理。按位运算符的含义如表 3-5 所示。

表 3-5　按位运算符的含义

按位运算符	名称	含义
&	按位和	例如$a&$b，表示对应位数都为 1，则结果改位为 1
|	按位或	例如$a\|$b，表示对应位数有一个为 1，则结果改位为 1
^	按位异或	例如$a^$b，表示对应位数不同，则结果改位为 1
~	按位取反	例如~$b，表示对应位数为 0 的改为 1，为 1 的改为 0
<<	左移	例如$a<<$b，表示将$a 在内存中二进制数据向左移动$b 位数，右边移空补 0
>>	右移	例如$a>>$b，表示将$a 在内存中二进制数据向右移动$b 位数，左边移空补 0

【例 3.20】（实例文件：ch03\3.20.php）

```
<?php
$a = 7;                                  // 7的二进制代码是 111
$b = 4;                                  // 4的二进制代码是 100
echo '$a & $b = ' . ($a & $b) . '<br/>';     // 运算结果为二进制代码100，即4
echo '$a | $b = ' . ($a | $b) . '<br/>';        // 运算结果为二进制代码111，即7
```

```
echo '$a ^ $b = '  . ($a ^ $b) . '<br/>';   // 运算结果为二进制代码011，即3
?>
```

本程序运行结果如图 3-20 所示。



图 3-20　程序运行结果

3.7.7　否定控制运算符

否定控制运算符用在“操作数”之前，用于对操作数值进行真假的判断。它包含一个逻辑否定运算符和一个按位否定运算符。否定控制运算符的含义如表 3-6 所示。

表 3-6　否定控制运算符的含义

否定控制运算符	含义
!	逻辑否
~	按位否

3.7.8　错误控制运算符

错误控制运算符用“@”来表示。在一个操作数之前使用，用来屏蔽错误信息的生成。

【例 3.21】（实例文件：ch03\3.21.php）

```
<?php
  $err = @(20 / 0) ;   // 如果不想显示这个错误，在表达式前加上“@”
?>
```

本程序运行结果如图 3-21 所示。



图 3-21　程序运行结果

3.7.9 三元运算符

三元运算符作用在三个操作数之间。这样的操作符在 PHP 中只有一个，即“？：”，语法形式如下：

```
(expr1) ? (expr2):(expr3)
```

如果 expr1 成立，则执行 expr2，否则执行 expr3。

【例 3.22】（实例文件：ch03\3.22.php）

```
<?php
$a = 5;
$b = 6;
echo ($a > $b) ? "大于成立" : "大于不成立"; echo "<br/>";   //大于不成立
echo ($a < $b) ? "小于成立" : "小于不成立"; echo "<br/>";  //小于成立
?>
```

本程序运行结果如图 3-22 所示。



图 3-22 程序运行结果

3.7.10 运算符的优先级和结合规则

运算符的优先级和结合其实与正常的数学运算符的规则十分相似。

- 加减乘除的先后顺序同数学运算中的完全一致。
- 对于括号，则先括号内再括号外。
- 对于赋值，则由右向左运行，即值依次从右边向左边的变量进行赋值。

3.8 表达式

表达式是在特定语言中表达一个特定的操作或动作的语句。PHP 的表达式也有同样的作用。

一个表达式包含“操作数”和“操作符”。操作数可以是变量，也可以是常量。操作符则体现了要表达的各个行为，如逻辑判断、赋值、运算等。

例如$a=5 就是表达式，而$a=5; 则为语句。另外，表达式也有值，例如$a=1 表达式的值为 1。



提 示

在 PHP 代码中，使用“;”号来区分表达式，即一个表达式和一个分号组成了一条 PHP 语句。在编写代码程序时，应该特别注意表达式后面的“;”，不要漏写或写错，否则会提示语法错误。

3.9 实战演练——创建多维数组

前面讲述了如何创建一维数组，本节讲述如何创建多维数组。多维数组和一维数组的区别是多维数组有两个或多个下标，它们的用法基本相似。

下面以创建二维数组为例进行讲解。

【例 3.23】（实例文件：ch03\3.23.php）

```
<?php
  $arr[0][0]=10;                    //定义数组元素并赋值
  $arr[0][1]=22;
  $arr[1] [0]= 1E+05;
  $arr[1] [1]= "开始学习 PHP 基本语法了";
  for ($i=0;$i<count($arr);$i++)        //使用 for 语句输出二维数组
 {
   for ($k=0;$k<count($arr[$i]);$k++)
   {
      $arr1=each($arr[$i]);
      echo "$arr1[value]<br />";
    }
 }
?>
```

本程序运行结果如图 3-23 所示。

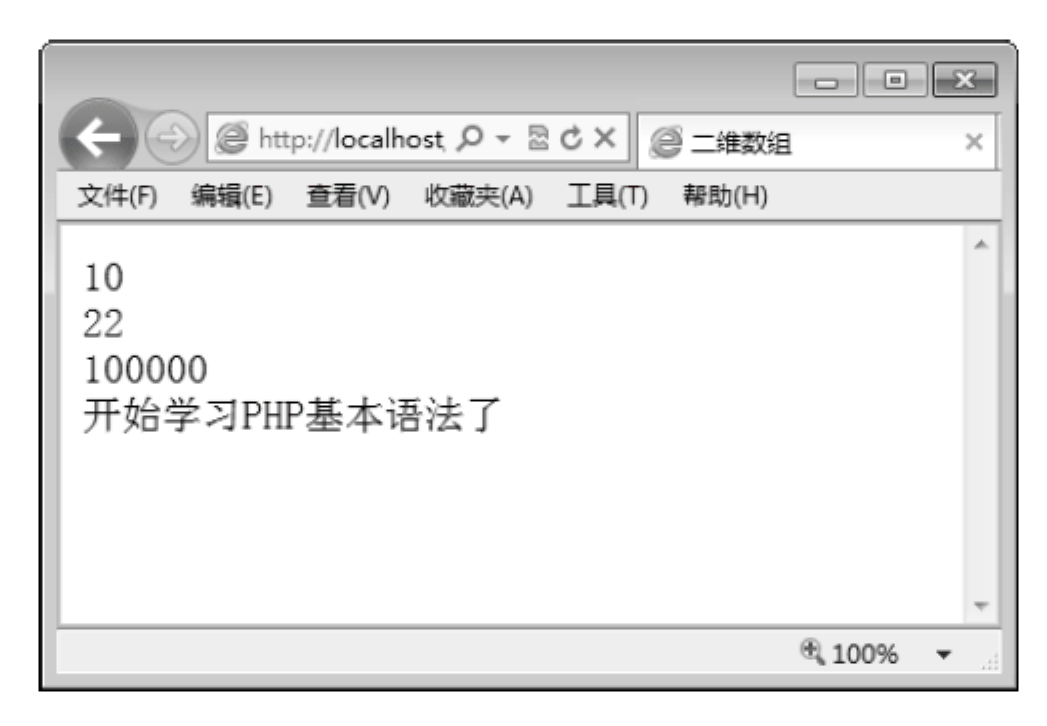


图 3-23 程序运行结果

3.10 高手私房菜

技巧 1：如何灵活运用命名空间（namespace）？

命名空间（namespace）作为一个比较宽泛的概念，可以理解为用来封装各个项目的方法。有点像是在文件系统中不同文件夹路径和文件夹当中的文件。两个文件的文件名可以完全相同，但是在不同的文件夹路径下，就是两个完全不同的文件。

PHP 的命名空间也是这样的一个概念。它主要用于在“类的命名”、“函数命名”及“常量命名”中避免代码冲突和在命名空间下管理变量名和常量名。

命名空间是使用 namespace 关键字在文件头部定义的，例如：

```
<?php
namespace 2ndbuilding\number24;  //命名空间
class room{}
$room = new __NAMESPACE__.room;
?>
```

命名空间还可以拥有子空间，就像文件夹的路径一样。可以通过内置变量_NAMESPACE_来使用命名空间及其子空间。

技巧 2：如何快速区分常量与变量？

常量和变量的明显区别如下。

- 常量前面没有美元符号（$）。
- 常量只能用 define() 函数定义，而不能通过赋值语句定义。
- 常量可以不用理会变量范围的规则而在任何地方定义和访问。
- 常量一旦定义就不能被重新定义或者取消定义。
- 常量的值只能是标量。

3.11　经典习题

（1）制作一个包含常量的例子，输出当前文件的路径。
（2）制作一个包含变量的例子，分析变量引用前和引用后的变化。
（3）制作一个包含数据类型之间的相互转换的例子。
（4）制作一个包含一维数组和多位数组的例子。

第 4 章　PHP 语言结构

在任何一种语言中，都有程序结构，常见的有顺序结构、分支结构和循环结构。在学习程序结构前，读者还需要对函数的知识进行学习。本章主要介绍 PHP 语言中函数和语言结构的使用方法和技巧。

本章学习目标

- 熟悉函数的使用方法
- 熟悉流程控制的概述
- 掌握条件控制结构
- 掌握循环控制结构
- 掌握条件分支结构的综合应用
- 掌握循环控制结构的综合应用

4.1　内置函数

函数的英文为 function，function 是功能的意思。顾名思义，使用函数就是要在编程过程中实现一定的功能，即通过代码块来实现一定的功能。比如通过一定的功能记录下酒店客人的个人信息，每到他生日的时候自动给他发送祝贺 email，并且这个发信“功能”可以重用，可更改为在某个客户的结婚纪念日时给他发送祝福 email。所以函数就是实现一定功能的一段特定的代码。

PHP 提供了大量的内置函数，方便程序员直接使用，常见的内置函数包括数学函数、字符串函数、时间和日期函数等。

下面以调用数学函数 rand()为例进行讲解。

【例 4.1】（实例文件：ch04\4.1.php）

```
<?php
  echo rand ()."<br />";                    //返回随机整合
  echo rand (1000,9999)."<br />";                    //产生一个4位随机整数
?>
```

本程序运行结果如图 4-1 所示。



图 4-1　程序运行结果

4.2 自定义函数

其实，更多的情况下，程序员需要的是自定义函数。

4.2.1 自定义和调用函数

自定义函数的语法结构如下：

```
function name_of_function( param1, param2, … ){
       statement
}
```

其中 name_of_function 是函数名，param1、param2 是参数，statement 是函数的具体内容。

下面以自定义和调用函数为例进行讲解。本实例主要实现酒店欢迎信息。

【例 4.2】（实例文件：ch04\4.2.php）

```
<?php
function sayhello($customer){            //自定义函数 sayhello
       return $customer.", 欢迎您来到 GoodHome 酒店。";
  }
  echo sayhello('张先生');                //调用函数 sayhello
?>
```

本程序运行结果如图 4-2 所示。

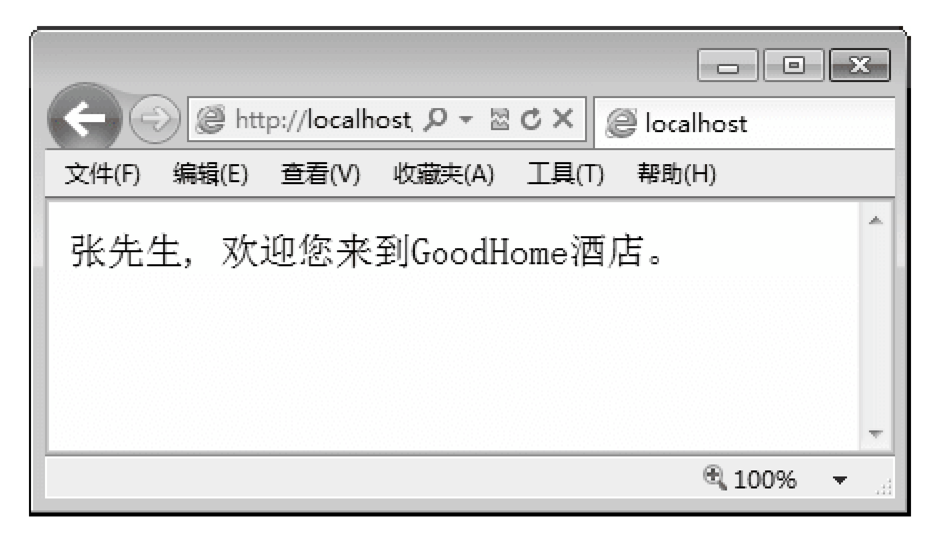


图 4-2 程序运行结果

值得一提的是，此函数的返回值是通过值返回的。也就是说 return 语句返回值时，创建了一个值的拷贝，并把它返回给使用此函数的命令或函数，在这里是 echo 命令。

4.2.2 向函数传递参数值

由于函数是一段封闭的程序，很多时候，程序员都需要向函数内传递一些数据来进行操作。

```
function 函数名称（参数1，参数2）{
       算法描述，其中使用参数1和参数2；
}
```

下面以计算酒店房间住宿费总价为例进行讲解。

【例 4.3】（实例文件：ch04\4.3.php）

```
<?php
```

```
    function totalneedtopay($days,$roomprice){        // 声明自定义函数
          $totalcost = $days*$roomprice;          // 计算住宿费总价
       echo  "需要支付的总价:$totalcost"."元。"; // 计算住宿费总价
    }
    $rentdays = 3;     //声明全局变量
    $roomprice = 168;
    totalneedtopay($rentdays,$roomprice);  //通过变量传递参数
    totalneedtopay(5,198);              //直接传递参数值
  ?>
```

运行结果如图 4-3 所示。

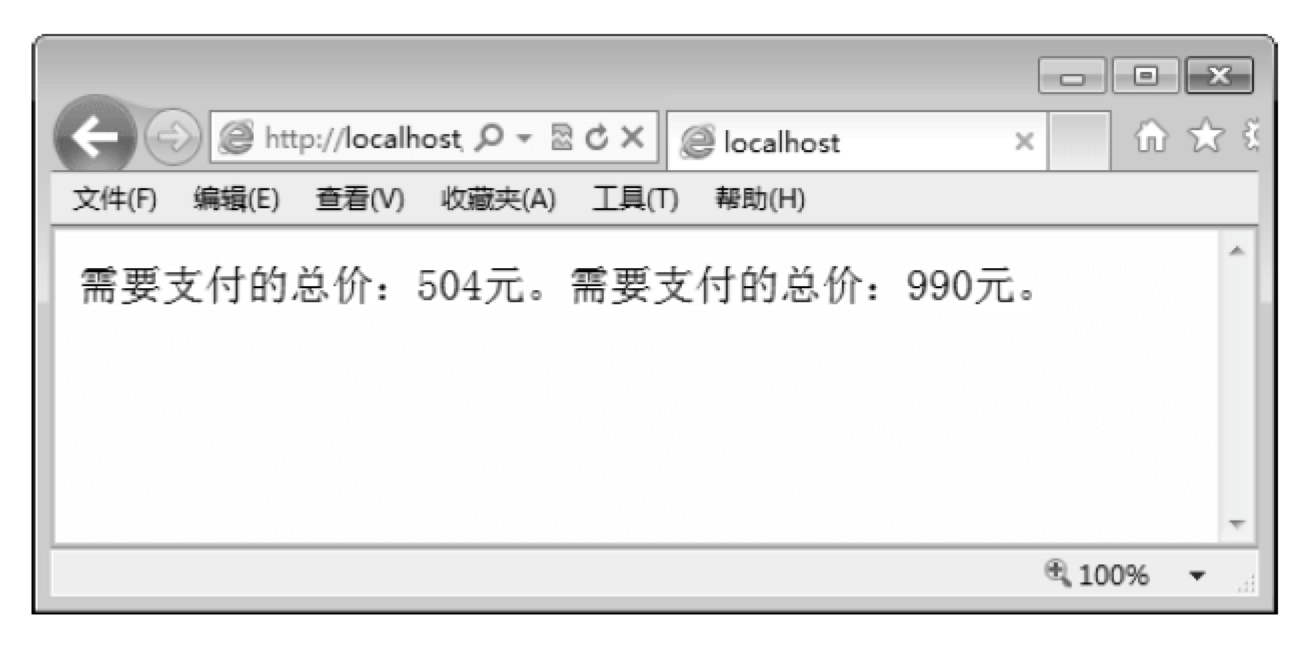


图 4-3　程序运行结果

【案例分析】：

（1）以这种方式传递参数值的方法就是向函数传递参数值。

（2）其中 function totalneedtopay($days,$roomprice){}定义了函数和参数。

（3）不管是通过变量$rentdays 和$roomprice 向函数内传递参数值，还是像 totalneedtopay (5,198)这样直接传递参数值都是一样的。

4.2.3　向函数传递参数引用

向函数传递参数引用，其实就是向函数传递变量引用。参数引用一定是变量引用，静态数值是没有引用一说的。由于在变量引用中已经知道，变量引用其实就是对变量名的使用，是对特定的变量位置的使用。

下面仍然以计算酒店服务费总价为例进行讲解。

【例 4.4】（实例文件：ch04\4.4.php）

```
  <?php
     $fee = 300;
    $serviceprice = 50;
    function totalfee(&$fee,$serviceprice){      // 声明自定义函数，参数前多了&，表示
按引用传递
          $fee = $fee+$serviceprice;          // 改变形参的值，实参的值也会发生改变
        echo "需要支付的总价:$fee"."元。";
    }
    totalfee($fee,$serviceprice);    //函数外部调用 fun()函数前$fee =300
```

```
  totalfee($fee,$serviceprice);    //函数外部调用 fun()函数后$ fee =350
?>
```

运行结果如图 4-4 所示。

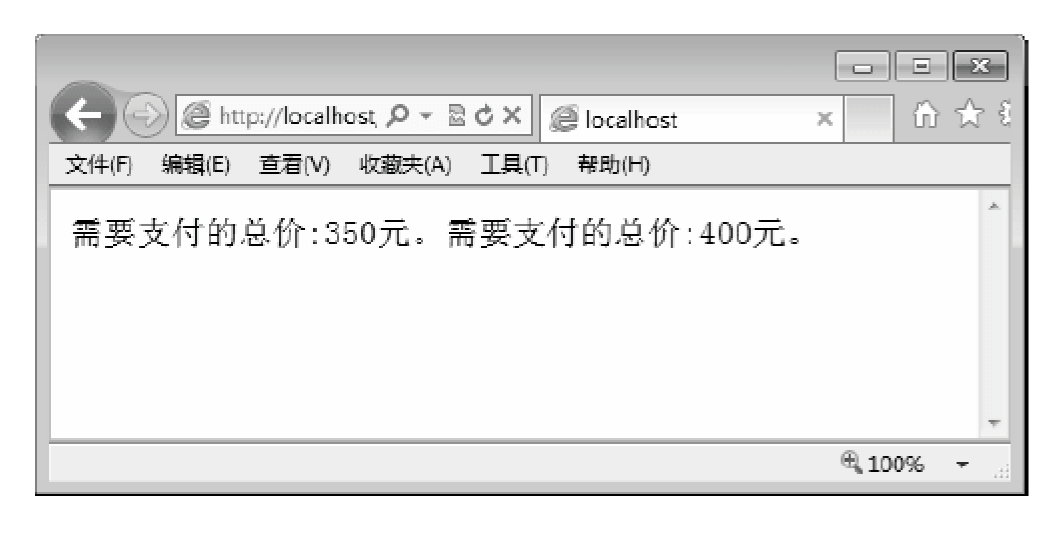


图 4-4 程序运行结果

【案例分析】：

（1）以这种方式传递参数值的方法就是向函数传递参数引用。使用“&”符号表示参数引用。

（2）其中 function totalfee(&$fee,$serviceprice){}定义了函数、参数和参数引用。变量$fee 是以参数引用的方式进入函数的。当函数的运行结果改变了变量$fee 的引用的时候，在函数外的变量$fee 的值也发生了改变，也就是函数改变了外部变量的值。

4.2.4 从函数中返回值

以上的例子中，都是把函数运算完成的值直接打印出来。但是，很多情况下，程序并不需要直接把结果打印出来，而是仅仅给出结果，并且把结果传递给调用这个函数的程序，为其所用。

这里需要用到 return 关键字。下面以综合酒店客房价格和服务价格为例进行讲解。

【例 4.5】（实例文件：ch04\4.5.php）

```
<?php
 function totalneedtopay($days,$roomprice){     // 声明自定义函数
        return $days*$roomprice;        // 返回酒店消费总价格
 }
  $rentdays = 3;
  $roomprice = 168;
  echo totalneedtopay($rentdays,$roomprice);
?>
```

运行结果如图 4-5 所示。



图 4-5 程序运行结果

【案例分析】:

（1）在函数 function totalneedtopay($days,$roomprice)算法中，直接使用 return 把运算的值返回给调用此函数的程序。

（2）其中 echo totalneedtopay($rentdays,$roomprice);语句调用了此函数，totalneedtopay()把运算值返回给了 echo 语句，才有上面的显示。当然这里也可以不用 echo 来处理返回值，也可以对它进行其他处理，比如赋值给变量等。

4.2.5　对函数的引用

不管是 PHP 中的内置函数，还是程序员在程序中自定义的函数，都可以简单地通过函数名调用。但是操作过程也有些不同，大致分为以下 3 种情况。

- 如果是 PHP 的内置函数，如 date()，可以直接调用。
- 如果这个函数是 PHP 的某个库文件中的函数，则需要用 include()或 require()命令把此库文件加载，然后才能使用。
- 如果是自定义函数，如果与引用程序同在一个文件中，则可直接引用。如果此函数不在当前文件内，则需要用 include()或 require()命令加载。

对函数的引用，实际上是对函数返回值的引用。

【例 4.6】（实例文件：ch04\4.6.php）

```
<?php
function &example($aa=1){                    //定义一个函数，别忘了加“&”符号
return $aa;                                  //返回参数$str
}
$bb= &example("引用函数的实例");              //声明一个函数的引用$str1;
echo $bb. "<br />";
?>
```

运行结果如图 4-6 所示。

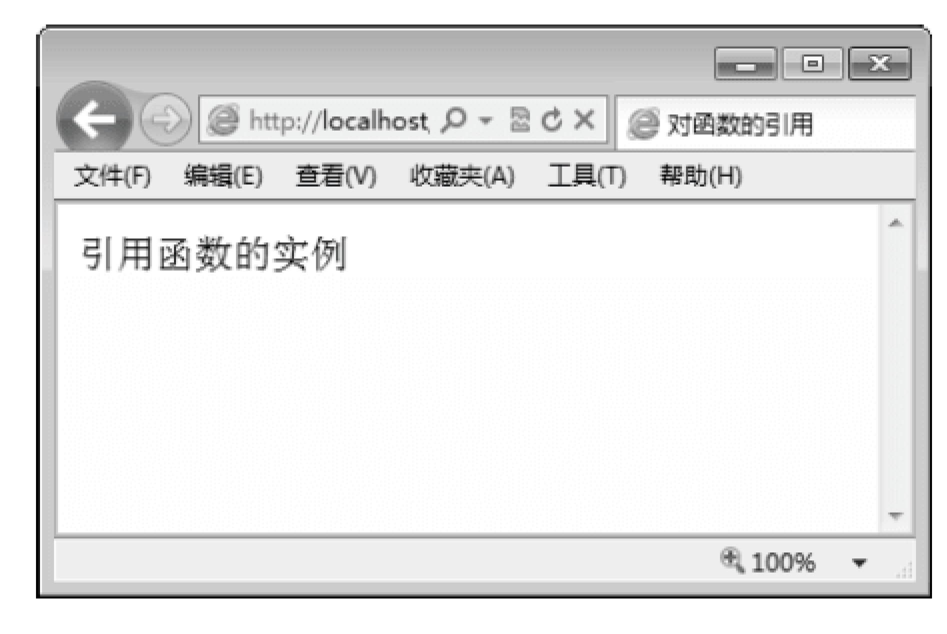


图 4-6　程序运行结果

【案例分析】:

（1）本实例首先定义一个函数，然后变量$bb 将引用函数，最后输出变量$bb，它实际上是$aa 的值。

（2）和参数传递不同，在定义函数和引用函数时，都必须使用“&”符号，表明返回的是一

个引用。

4.2.6 对函数取消引用

对于不需要引用的函数，可以做取消操作。取消引用使用 unset()函数来完成，目的是断开变量名和变量内容之间的绑定，此时并没有销毁变量内容。

【例 4.7】（实例文件：ch04\4.7.php）

```
<?php
 $num = 166;                          //声明一个整型变量
 $math = &$num;                          //声明一个对变量$num 的引用$math
 echo "\$math is:  ".$math."<br />";          //输出引用$math
 unset($math);                           //取消引用$math
 echo "\$num is:  ".$num;                    //输出原变量
?>
```

运行结果如图 4-7 所示。



图 4-7 程序运行结果

本程序首先声明一个变量和变量的引用，输出引用后取消引用，再次调用原变量。从图 4-6 可以看出，取消引用后对原变量没有任何影响。

4.3 包含文件

如果想让自定义的函数被多个文件使用，可以将自定义函数组织到一个或者多个文件中，这些收集函数定义的文件就是用户自己创建的 PHP 函数库。通过使用 require ()和 include()等函数可以将函数库载入到脚本程序中。

4.3.1 require 和 include

require()和 include()语句不是真正意义的函数，属于语言结构。通过 include()和 require()语句都可以实现包含并运行指定文件。

（1）require()：在脚本执行前读入它包含的文件，通常在文件的开头和结尾处使用。

（2）include()：在脚本读到它的时候才将包含的文件读进来，通常在流程控制的处理区使用。

require()和 include()语句对于处理失败方面是不同的。当文件读取失败后，requie 将产生一个

致命错误，而 include 则产生一个警告。可见，如果遇到文件丢失时需要继续运行，则使用 include，如果想停止处理页面，则使用 requie。

【例 4.8】（实例文件：ch04\4.8.php 和 test.php）

其中，4.8.php 代码如下：

```
<?php
$aaa = '杨柳青青江水平';      //定义一个变量 aaa
$bbb = '闻郎江上唱歌声';     //定义一个变量 bbb
?>
```

test.php 代码如下：

```
<?php
echo " $aaa $bbb";   //未载入文件前调用两个变量
include '4.8.php';
echo " $aaa $bbb ";  //载入文件后调用两个变量
?>
```

运行 test.php 结果如图 4-8 所示。从结果可以看出，使用 include 时，虽然出现了警告，但是脚本程序仍然在运行。

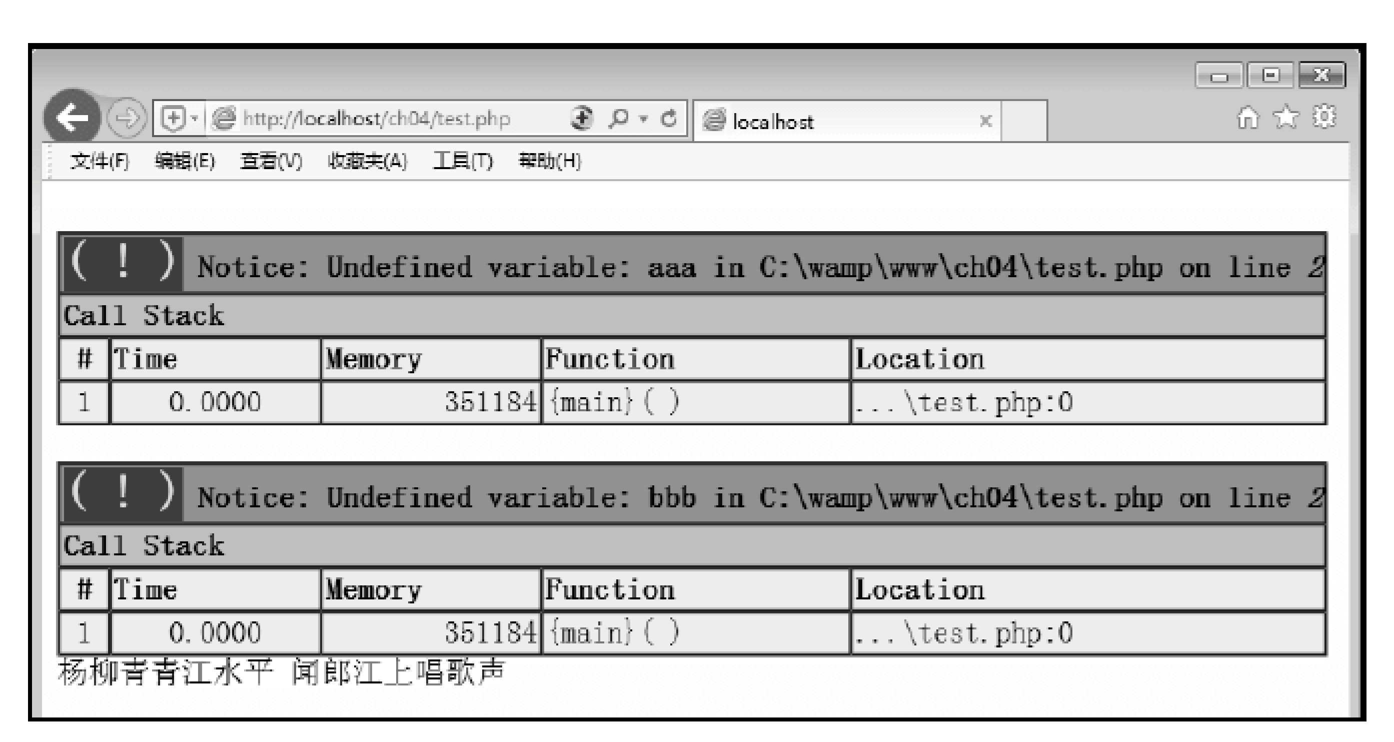


图 4-8　程序运行结果

4.3.2　include_once 和 require_once

include_once 和 require_once 语句在脚本执行期间包含并运行指定文件，作用与 include 和 require 语句类似，唯一的区别是，如果该文件的代码被包含了，则不会再次包含，只会包含一次。从而避免函数重定义以及变量重赋值等问题。

4.4　流程控制概述

流程控制，也叫控制结构，在一个应用中用来定义执行程序流程。它决定了某个程序段是否会被执行和执行多少次。

PHP 中的控制语句分为 3 类：顺序控制语句、条件控制语句和循环控制语句。其中顺序控制语句是从上到下依次执行的，这种结构没有分支和循环，是 PHP 程序中最简单的结构。下面主要

讲述条件控制语句和循环控制语句。

4.5 条件控制结构

条件控制语句中包含两个主要的语句，一个是 if 语句，一个是 switch 语句。

4.5.1 单一条件分支结构（if 语句）

if 语句是最为常见的条件控制语句，它的格式为：

```
if（条件判断语句）{
        命令执行语句;
}
```

这种形式只是对一个条件进行判断。如果条件成立，则执行命令语句，否则不执行。

if 语句的流程控制图如图 4-9 所示。

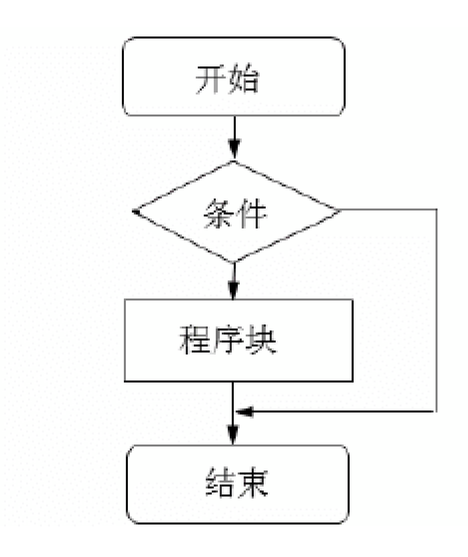


图 4-9 if 语句流程控制图

【例 4.9】（实例文件：ch04\4.9.php）

```
<?php
 $num = rand(1,100);                    //使用 rand()函数生成一个随机数
 if ($num % 2 != 0){                    //判断变量$num 是否为奇数
     echo "\$num = $num";                   //如果为奇数，输出表达式和说明文字
     echo "<br />$num 是奇数。";
 }
?>
```

运行后刷新页面，结果如图 4-10 所示。



图 4-10 程序运行结果

【案例分析】：

（1）此实例首先使用 rand()函数随机生成一个整数$num，然后判断这个随机整数是不是奇数，

如果是，则输出上述结果，如果不是，则不输出任何内容，所以如果页面内容显示为空，则刷新页面即可。

（2）rand() 函数返回随机整数，语法格式如下：

```
rand(min,max)
```

此函数主要是返回 min 和 max 之间的一个随机整数。如果没有提供可选参数 min 和 max，则 rand() 返回 0 到 RAND_MAX 之间的伪随机整数。

4.5.2 双向条件分支结构（if…else 语句）

如果是非此即彼的条件判断，可以使用 if…else 语句。它的格式为：

```
if（条件判断语句）{
      命令执行语句 A;
}else{
      命令执行语句 B;
}
```

这种结构形式首先判断条件是否为真，如果为真，则执行命令语句 A，否则执行命令语句 B。if…else 语句程序控制流程图如图 4-11 所示。

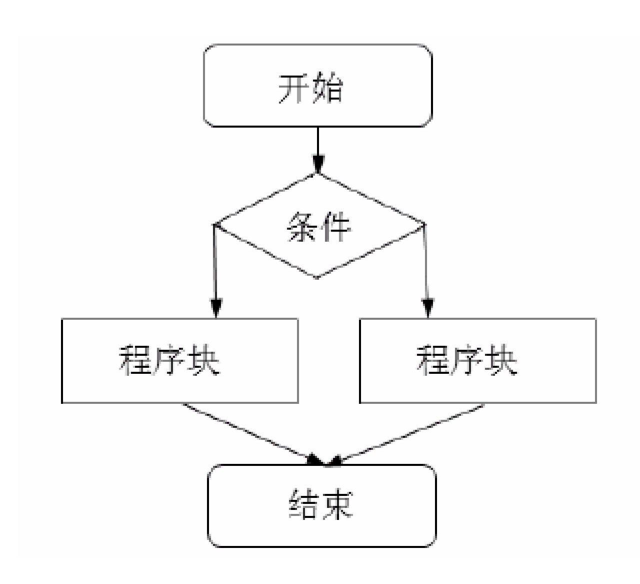


图 4-11 if…else 语句控制流程图

【例 4.10】（实例文件：ch04\4.10.php）

```
<html>
<?php
$d=date("D");                //定义时间变量
if ($d=="Fri")                //判断时间变量是否等于周五
  echo "今天是周五哦!";
else
  echo "可惜今天不是周五!";
?>
```

运行后结果如图 4-12 所示。

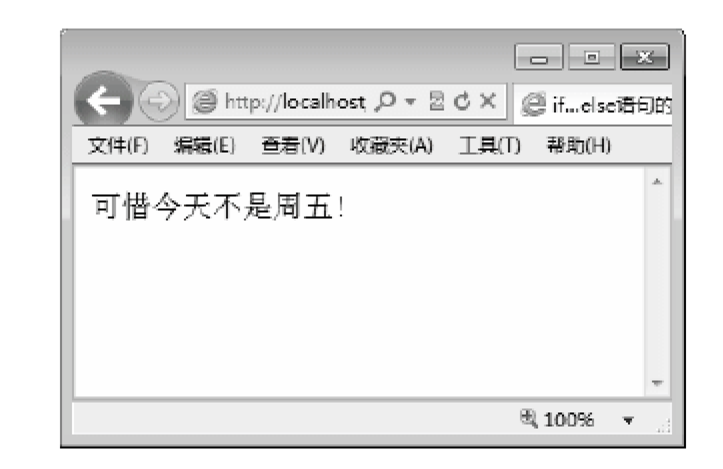


图 4-12　程序运行结果

4.5.3　多向条件分支结构（elseif 语句）

在条件控制结构中，有时会出现多于两种的选择，此时可以使用 elseif 语句。它的语法格式为：

```
    if（条件判断语句）{
            命令执行语句;
}elseif（条件判断语句）{
            命令执行语句;
}…
 else{
            命令执行语句;
}…
```

elseif 语句程序控制流程图如图 4-13 所示。

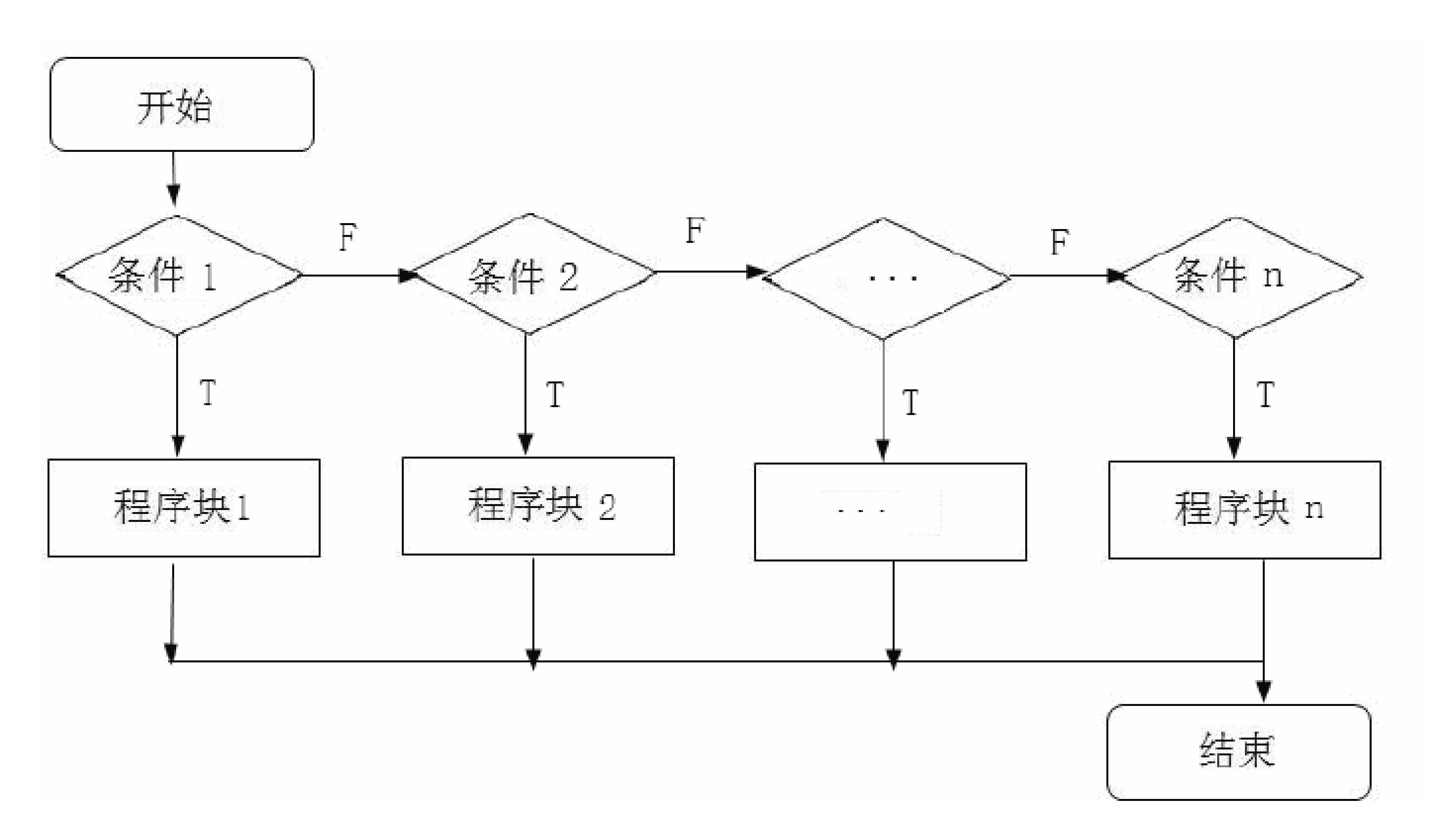


图 4-13　elseif 语句控制流程图

【例 4.11】（实例文件：ch04\4.11.php）

```
<?php
$score = 85;                              //设置成绩变量$score
if ($score >= 0 and $score <= 60){         //判断成绩变量是否在0~60之间
echo "您的成绩为差";                      //如果是，说明成绩为差
}elseif($score > 60 and $score <= 80){ //否则判断成绩变量是否在61~80之间
echo "您的成绩为中等";                  //如果是，说明成绩为中等
}else{                                   //如果两个判断都是 false，则输出默认值
echo "您的成绩为优等";                  //说明成绩为优等
}
```

```
?>
```

运行后结果如图 4-14 所示。



图 4-14 程序运行结果

4.5.4 多向条件分支结构（switch 语句）

switch 语句的结构给出不同情况下可能执行的程序块，条件满足哪个程序块，就执行哪个语句。它的语法格式为：

```
switch（条件判断语句）{
          case 可能判断结果 a：
                命令执行语句；
break;
           case 可能判断结果 b：
                命令执行语句；
break;
           …
           default：
                命令执行语句；
}
```

其中，若“条件判断语句”的结果符合某个“可能判断结果”，就执行其对应的“命令执行语句”。如果都不符合，则执行 default 对应的默认项的“命令执行语句”。

switch 语句的流程控制图如图 4-15 所示。

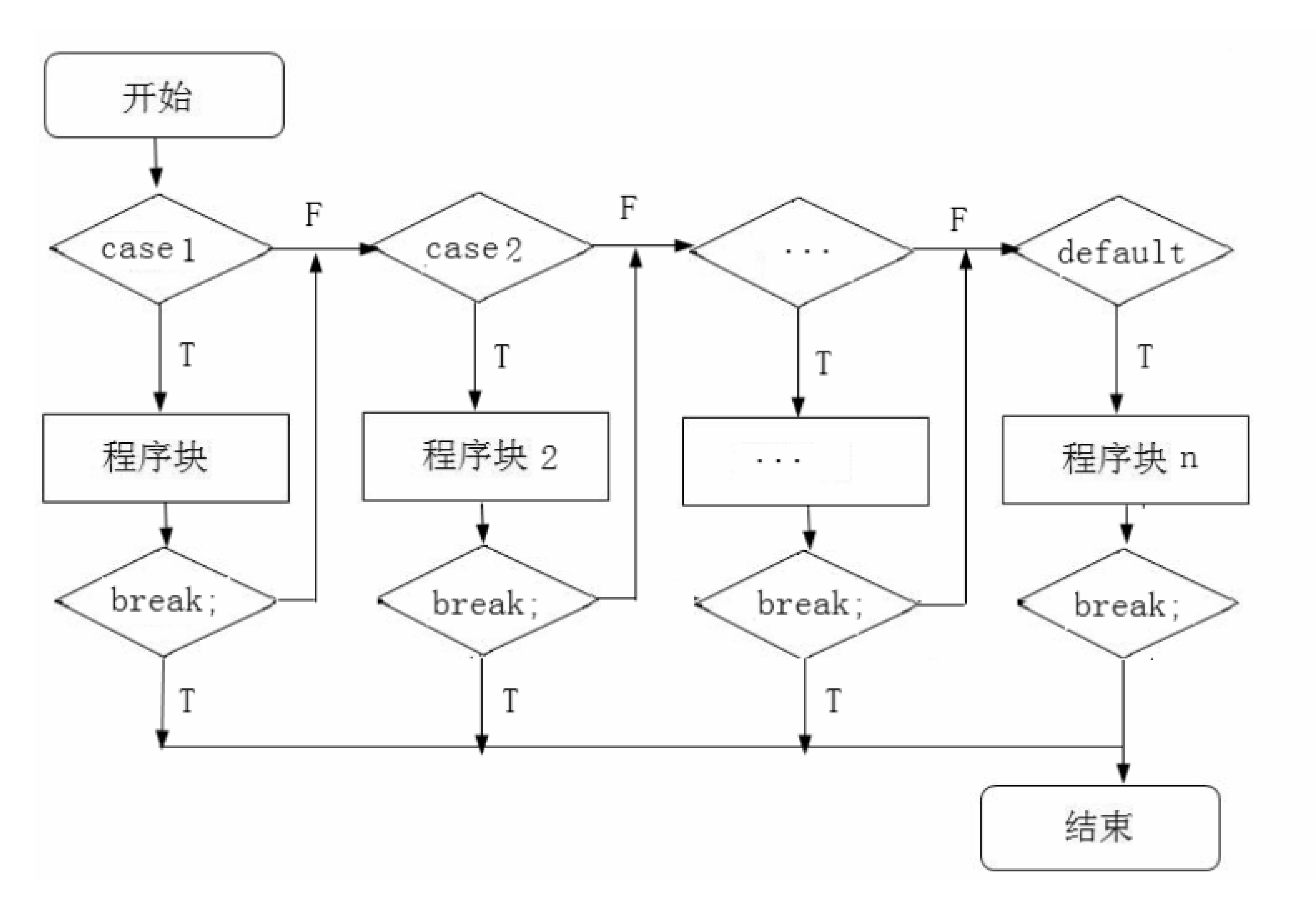


图 4-15 switch 语句控制流程图

【例 4.12】（实例文件：ch04\4.12.php）

```
<?php
$x=5;           //定义变量$x
switch ($x)      //判断$x 与1~5之间的关系
{
case 1:
  echo "数值为 1";
  break;
case 2:
  echo "数值为2";
  break;
case 3:
  echo "数值为3";
  break;
case 4:
  echo "数值为4";
  break;
case 5:
  echo "数值为5";
  break;
default:
  echo "数值不在1到5之间";
}
?>
```

运行后结果如图 4-16 所示。



图 4-16　程序运行结果

4.6　循环控制结构

循环控制语句主要包括三种，即 while 循环、do…while 循环和 for 循环。while 循环在代码运行的开始检查表述的真假；而 do…while 循环则在代码运行的末尾检查表述的真假，这样，do…while 循环至少要运行一遍。

4.6.1　while 循环语句

while 循环的结构为：

```
while （条件判断语句）{
     命令执行语句;
```

```
    }
```

其中当“条件判断语句”为 true 时，执行后面的“命令执行语句”，然后返回到条件表达式继续进行判断，直到表达式的值为假，才能跳出循环，执行后面的语句。

while 循环语句的流程控制图如图 4-17 所示。

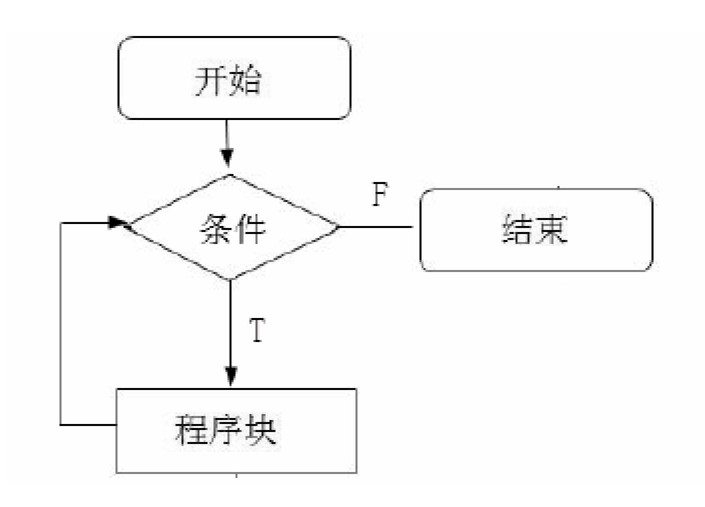


图 4-17　while 语句控制流程图

【例 4.13】（实例文件：ch04\4.13.php）

```
<?php
 $num = 1;                   //定义变量$num
 $str = "20以内的奇数为："；  //定义变量$x
 while($num <=20){            //判断$num 是否小于或等于20
     if($num % 2!= 0){        //判断$num 是否为奇数，为奇数则输出，否则做加一操作
         $str .= $num." ";
     }
     $num++;
 }
 echo $str;
?>
```

运行后结果如图 4-18 所示。

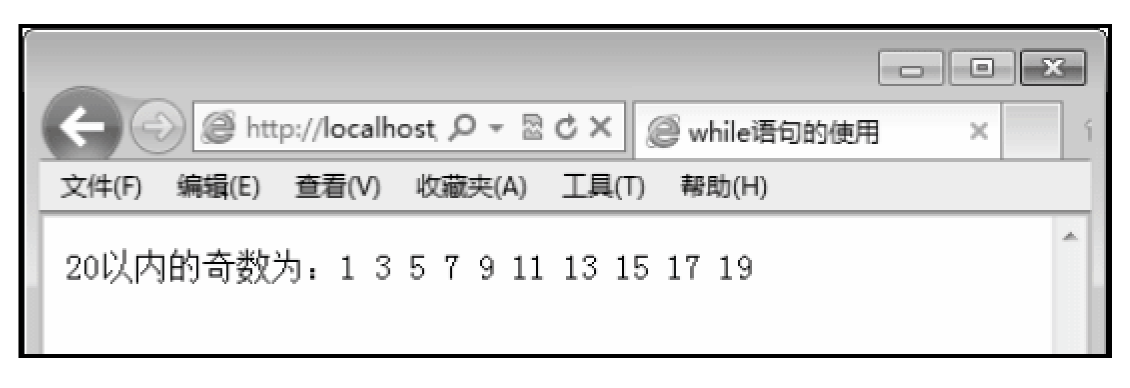


图 4-18　程序运行结果

本实例主要实现 20 以内的奇数输出。从 1~20 依次判断是否为奇数，如果是，则输出；如果不是，则继续下一次的循环。

4.6.2　do…while 循环语句

do…while 循环的结构为：

```
do{
   命令执行语句;
}while (条件判断语句)
```

其中先执行 do 后面的“命令执行语句”，其中的变量会随着命令的执行发生变化。当此变量

通过 while 后的“条件判断语句”判断为 false 时，停止执行“命令执行语句”。

do…while 循环语句的流程控制图如图 4-19 所示。

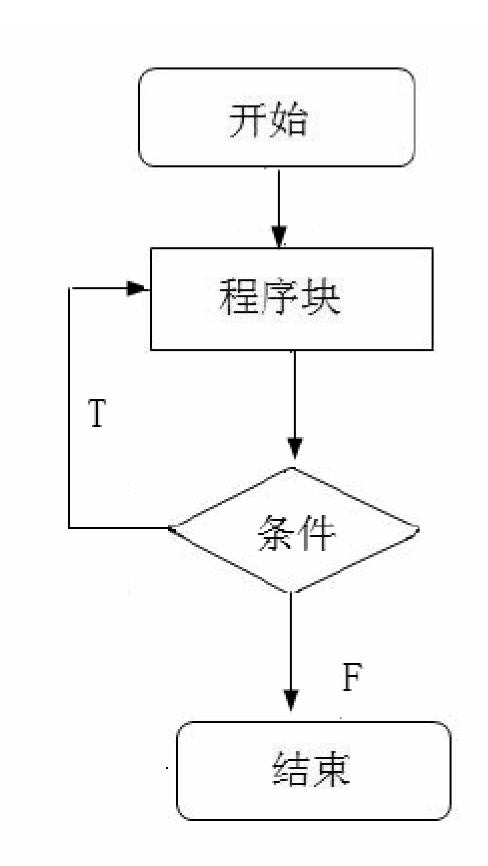


图 4-19　do…while 循环语句控制流程图

【例 4.14】（实例文件：ch04\4.14.php）

```
<?php
 $aa = 0;                                          //声明一个整数变量$aa
 while($aa != 0){                                    //使用 while 循环输出
     echo "不会被执行的内容";                        //这句话不会被输出
 }
 do{                                               //使用 do…while 循环输出
     echo "被执行的内容";                            //这句话会被输出
 }while($aa != 0);
?>
```

运行后结果如图 4-20 所示。从结果可以看出，while 语句和 do…while 语句有很大的区别。

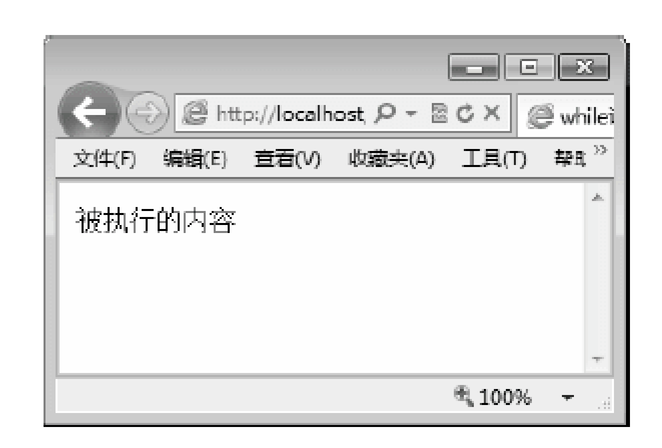


图 4-20　程序运行结果

4.6.3　for 循环语句

for 循环的结构为：

```
for (expr1; expr2; expr3)
{
执行命令语句
 }
```

其中 expr1 为条件的初始值，expr2 为判断的最终值，通常都使用比较表达式或逻辑表达式充当判断的条件，执行完命令语句后，再执行 expr3。

for 循环语句的流程控制图如图 4-21 所示。

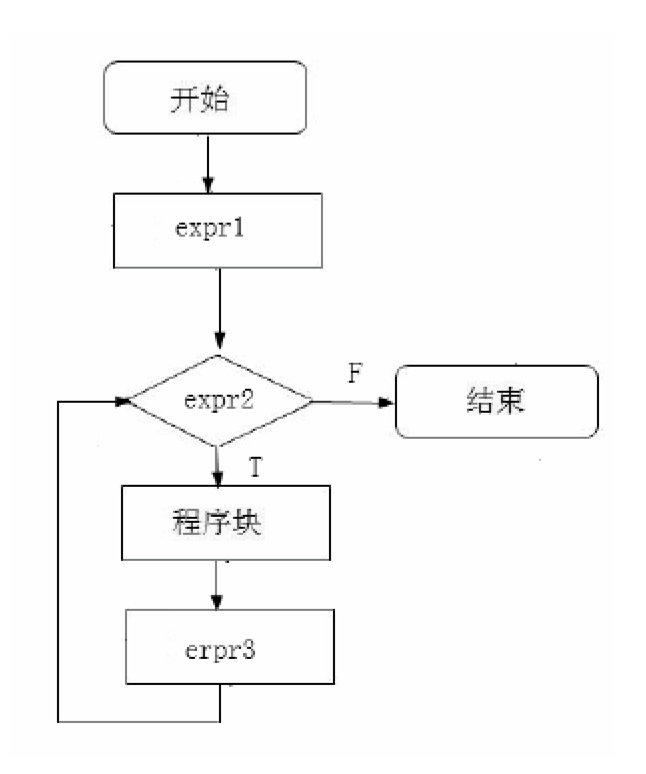


图 4-21　for 循环语句控制流程图

【例 4.15】（实例文件：ch04\4.15.php）

```
<?php
for($i=0;$i<4;$i++){                //使用 for 循环输出
   echo "for 语句的功能非常强大<br/>";
}
?>
```

运行结果如图 4-22 所示。从图中可以看出，命令语句执行了 4 次。

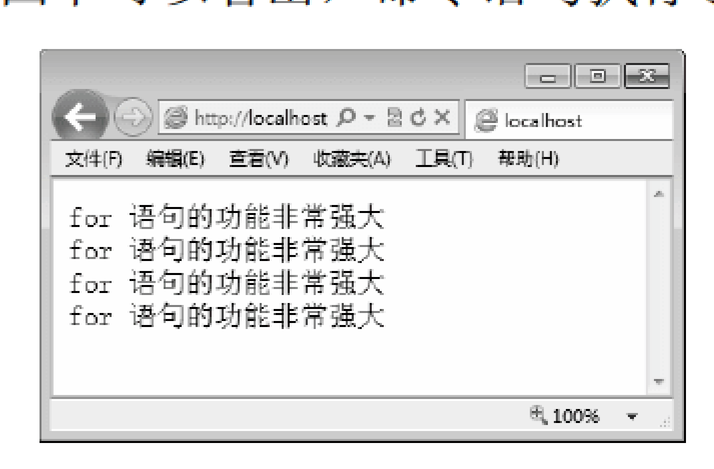


图 4-22　程序运行结果

4.6.4　foreach 循环语句

foreach 语句是常用的一种循环语句，它经常被用来遍历数组元素。它的格式为：

```
foreach（数组 as 数组元素）{
        对数组元素的操作命令；
}
```

可以根据数组的情况分为两种，即不包含键值的数组和包含键值的数组。

不包含键值的：

```
foreach（数组 as 数组元素值）{
        对数组元素的操作命令；
}
```

包含键值的：

```
foreach (数组 as 键值 => 数组元素值) {
        对数组元素的操作命令;
}
```

每进行一次循环，当前数组元素的值就会被赋值给数组元素值变量，数组指针会逐一地移动，直到遍历结束为止。

【例 4.16】（实例文件：ch04\4.16.php）

```
<?php
$arr=array("one", "two", "three");
foreach ($arr as $value)    //使用 foreach 循环输出
{
    echo "数组值: " . $value . "<br />";
}
?>
```

运行结果如图 4-23 所示。从图中可以看出，命令语句执行了 3 次。

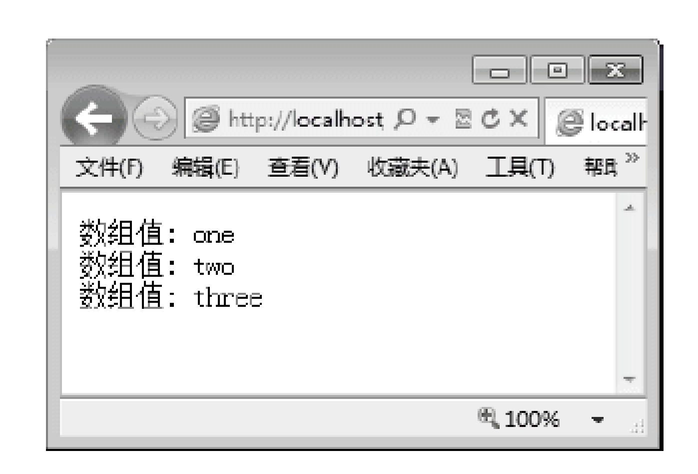


图 4-23　程序运行结果

4.6.5　流程控制的另一种书写格式

在一个含有多条件、多循环的语句中，包含多个“{}”，查看起来比较烦琐。流程控制语言的另外一种书写方式是以“:”来代替左边的大括号，使用 endif;、endwhile、endfor、endforeach;和 endswitch;来替代右边的大括号，这种描述程序结构的可读性比较强。常见的格式如下。

条件控制语句中的 if 语句：

```
if(条件判断语句):
    命令执行语句;
elseif(条件判断语句):
    命令执行语句;
elseif(条件判断语句):
    命令执行语句;
…
else:

    命令执行语句;
endif;
```

条件控制语句中的 switch 语句：

```
switch(条件判断语句):
     case  可能结果 a:
      命令执行语句;
 case  可能结果 b:
      命令执行语句;
    …
 default:
      命令执行语句;
endswitch;
```

循环控制语句中的 while 循环：

```
while(条件判断语句):
  命令执行语句
endwhile;
```

循环控制语句中的 do…while 循环：

```
do
   命令执行语句
while (条件判断语句);
```

循环控制语句中的 for 循环：

```
for (起始表述; 为真的布尔表述; 增幅表述):
命令执行语句
endfor;
```

【例 4.17】（实例文件：ch04\4.17.php）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>杨辉三角</title>
</head>
<body>
<?php
 $mixnum = 1;
 $maxnum = 10;
 $tmparr[][] = array();
 $tmparr[0][0] = 1;
 for($i = 1; $i < $maxnum; $i++):
     for($j = 0; $j <= $i; $j++):
         if($j == 0 or $j == $i):
             $tmparr[$i][$j] = 1;
                else:
             $tmparr[$i][$j] = $tmparr[$i - 1][$j - 1] + $tmparr[$i - 1][$j];
         endif;
     endfor;
```

```
    endfor;
    foreach($tmparr as $value):
        foreach($value as $vl)
            echo $vl.' ';
        echo '<p>';
    endforeach;
?>
</body>
</html>
```

运行结果如图 4-24 所示。从效果图可以看出，该代码使用新的书写格式实现了杨辉三角的排列输出。



图 4-24　程序运行结果

4.6.6　使用 break/continue 语句跳出循环

使用 break 关键字，用来跳出（也就是终止）循环控制语句和条件控制语句中的 switch 语句的执行。例如：

```
<?php
$n = 0;
while (++$n) {
    switch ($n) {
    case 1:
        echo "case one";
        break ;
    case 2:
        echo "case two";
        break 2;
    default:
        echo "case three";
        break 1;
    }
}
?>
```

在这段程序中，while 循环控制语句里面包含一个 switch 流程控制语句。在程序执行到 break 语句时，break 会终止执行 switch 语句，或者是 switch 和 while 语句。其中在 “case 1” 下的 break 语句跳出 switch 语句。“case 2”下的 break 2 语句跳出 switch 语句和包含 switch 的 while 语句。“case 3”下的 break 1 语句和 “case 1” 下的 break 语句一样，只是跳出 switch 语句。其中，break 后带的数字参数是指 break 要跳出的控制语句结构的层数。

使用 continue 关键字的作用是，跳开当前的循环迭代项，直接进入到下一个循环迭代项，继续执行程序。下面通过一个实例说明此关键字作用。

【例 4.18】（实例文件：ch04\4.18.php）

```
<?php
$n = 0;
while ($n++ < 6) {   //使用 while 循环输出
   if ($n == 2){
        continue;
   }
    echo $n."<br />";
}
?>
```

运行结果如图 4-25 所示。

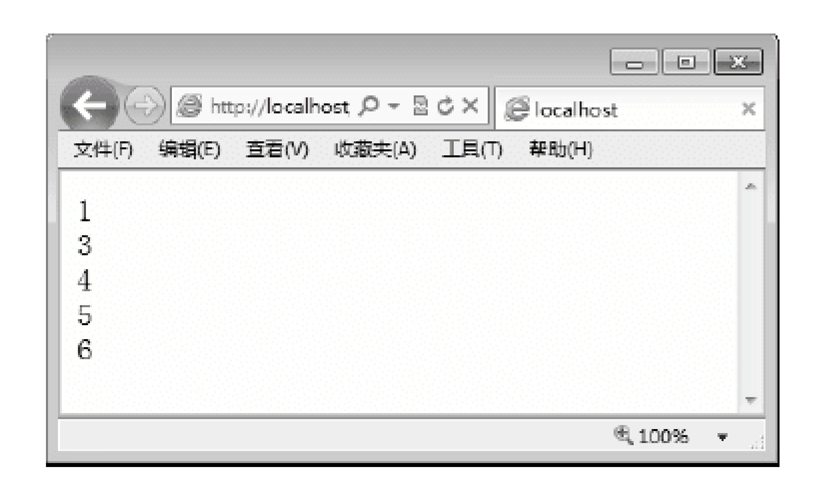


图 4-25　程序运行结果

其中 continue 关键字，在当 n 等于 2 的时候，跳出本次循环，并且直接进入到下一个循环迭代项，即当 n 等于 3。另外，continue 关键字和 break 关键字一样，都可以在后面直接跟一个数字参数，用来表示跳开循环的结构层数。“continue” 和 “continue 1” 相同，“continue 2” 表示跳开所在循环和上一级循环的当前迭代项。

4.7　实战演练 1——条件分支结构综合应用

下面，通过案例讲述条件分支结构的综合应用。

【例 4.19】（实例文件：ch04\4.19.php）

```
<?php
  $members = Null;
  function checkmembers($members){
    if ($members < 1){
      echo "我们不能为少于一人的顾客提供房间。<br />";
```

```
      }else{
        echo "欢迎来到 GoodHome 酒店。<br />";
      }
    }
    checkmembers(2);
    checkmembers(0.5);
    function checkmembersforroom($members){
      if ($members < 1){
        echo "我们不能为少于一人的顾客提供房间。<br />";
      }elseif( $members == 1 ){
        echo "欢迎来到 GoodHome 酒店。 我们将为您准备单床房。<br />";
      }elseif( $members == 2 ){
        echo "欢迎来到 GoodHome 酒店。 我们将为您准备标准间。<br />";
      }elseif( $members == 3 ){
        echo "欢迎来到 GoodHome 酒店。 我们将为您准备三床房。<br />";
      }else{
      echo "请直接电话联系我们，我们将依照具体情况为您准备合适的房间。<br />";
    }
    }
    checkmembersforroom(1);
    checkmembersforroom(2);
    checkmembersforroom(3);
    checkmembersforroom(5);
    function switchrooms($members){
      switch ($members){
              case  1:
                echo "欢迎来到 GoodHome 酒店。 我们将为您准备单床房。<br />";
          break;
              case  2:
                echo "欢迎来到 GoodHome 酒店。 我们将为您准备标准间。<br />";
          break;
              case  3:
                echo "欢迎来到 GoodHome 酒店。 我们将为您准备三床房。<br />";
          break;
              default:
              echo "请直接电话联系我们，我们将依照具体情况为您准备合适的房间。";
          break;
            }
     }
    switchrooms(1);
    switchrooms(2);
    switchrooms(3);
    switchrooms(5);
  ?>
```

运行结果如图 4-26 所示。

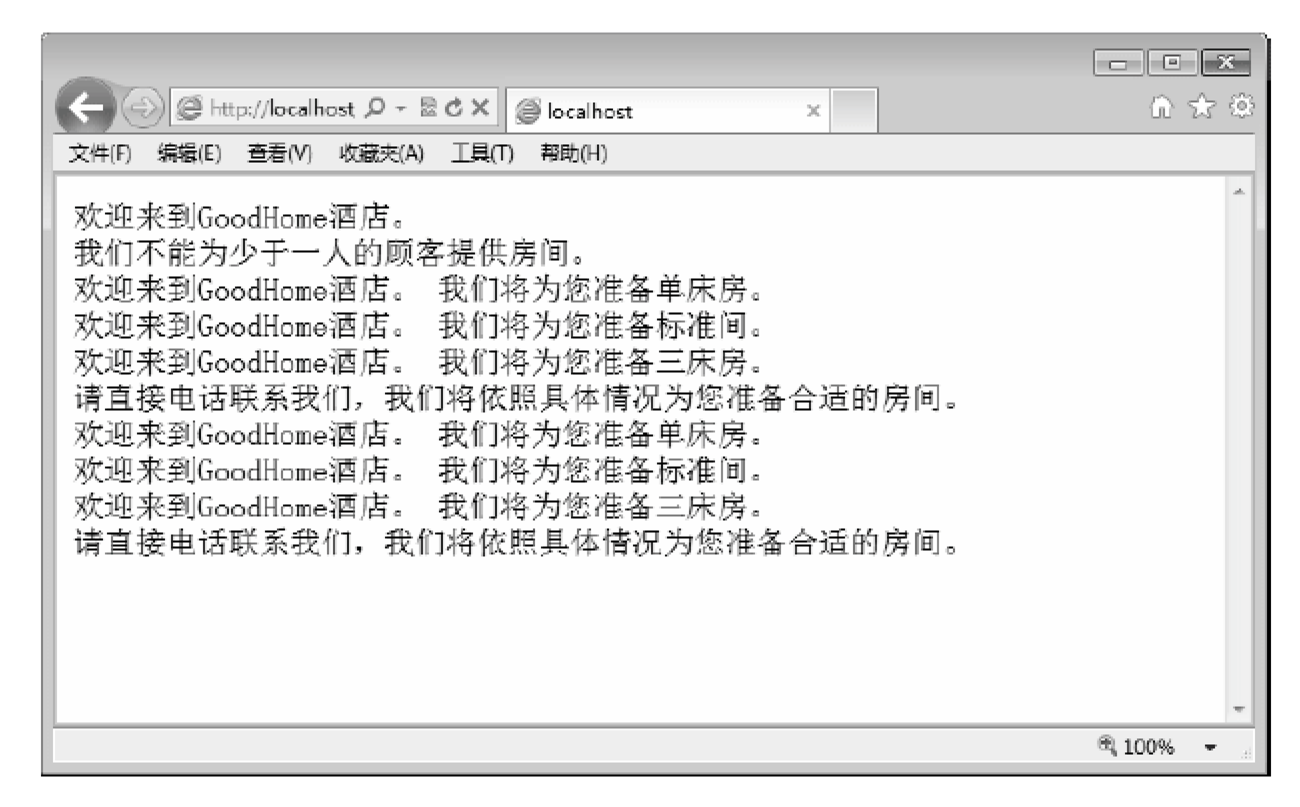


图 4-26　程序运行结果

其中最后 4 行由 switch 语句实现。其他输出均由 if 语句实现。

4.8　实战演练 2——循环控制结构综合应用

下面以遍历已订房间门牌号为例介绍循环控制语句的应用技巧。

【例 4.20】（实例文件：ch04\4.20.php）

```
<?php
  $bookedrooms = array('102','202','203','303','307');   //定义数组 bookedrooms
  for ($i = 0; $i < 5; $i++){                            //循环输出数组 bookedrooms
         echo $bookedrooms[$i]."<br />";
  }

  function checkbookedroom_while($bookedrooms){  //定义函数
       $i = 0;
     while (isset($bookedrooms[$i])){
    echo $i.":".$bookedrooms[$i]."<br />";
    $i++;
   }
  }
  checkbookedroom_while($bookedrooms);
  $i = 0;
  do{
   echo $i."-".$bookedrooms[$i]."<br />";
   $i++;
   }while($i < 2);
?>
```

运行结果如图 4-27 所示。

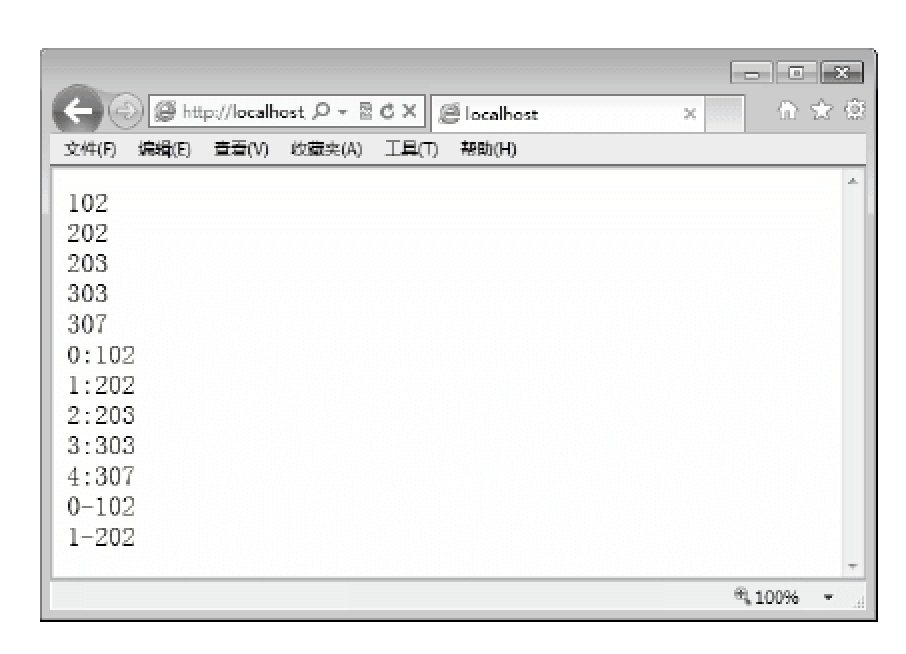


图 4-27　程序运行结果

其中，102~307 由 for 循环实现。0：102~4：307 由 while 循环实现。0-102 和 1-202 由 do…while 循环实现。for 循环和 while 循环都完全遍历了数组$bookedrooms，而 do…while 循环由于 while（$i < 2），所以 do 后面的命令执行了两次。

4.9　高手私房菜

技巧 1：如何合理运用 include_once()和 require_once()？

答：include()和 require()函数在其他 PHP 语句执行之前运行，引入需要的语句并加以执行。但是每次运行包含此语句的 PHP 文件时，include()和 require()函数都要运行一次。include()和 require()函数如果在先前已经运行过，并且引入相同的文件，则系统就会重复引入这个文件，从而产生错误。而 include_once()和 require_once()函数只是在此次运行的过程中引入特定的文件或代码，但是在引入之前，会先检查所需文件或者代码是否已经引入，如果引入将不再重复引入，从而不会造成冲突。

技巧 2：程序检查后正确，却显示 Notice: Undefined variable，为什么？

PHP 默认配置会报这个错误，这就是警告将在页面上打印出来，虽然这有利于暴露问题，但现实使用中会存在很多问题。通用解决办法是修改 php.ini 的配置，需要修改的参数如下：

（1）找到 error_reporting = E_ALL

修改为 error_reporting = E_ALL & ~E_NOTICE

（2）找到 register_globals = Off

修改为 register_globals = On

4.10　经典习题

（1）制作一个包含内置函数的例子，随机输出 10~99 之间的整数。

（2）制作一个包含自定义函数的例子，包含向函数传递参数引用。

（3）制作一个包含文件引用的例子。

（4）制作一个包含条件分支结构的例子。

（5）制作一个包含循环控制结构的例子。

第 5 章　字符串和正则表达式

字符串在 PHP 程序中经常被使用，那么如何格式化字符串、如何连接分离字符串、如何比较字符串等，是初学者经常遇到的问题。另外，本章还将讲述正则表达式的使用方法和技巧。

本章学习目标

- 掌握字符串单引号和双引号的使用
- 掌握字符串连接符的使用
- 掌握字符串的基本操作
- 熟悉正则表达式的基本概念
- 掌握正则表达式的语法规则

5.1　字符串的单引号和双引号

字符串是指一连串不中断的字符。这里的字符主要包括以下几种类型：

- 字母类型，如常见的 a、b、c 等。
- 数字类型，如常见的 1、2、3、4 等。
- 特殊字符类型，如常见的#、%、^、$等。
- 不可见字符类型，如回车符、Tab 字符和换行符等。

标识字符串通常使用单引号或双引号，表面看起来没有什么区别。但是，对于存在于字符串中的变量，这两个是不一样的。

（1）双引号内会输出变量的值。单引号内直接显示变量名称。

（2）双引号中可以通过“\”转义符输出的特殊字符如下。

- \n：新一行。
- \t：TAB。
- \\：反斜杠。
- \0：ASCⅡ码的 0。
- \$：把此符号转义为单纯的美元符号，而不再作为声明变量的标识符。
- \r：回车
- \{octal #}：八进制转义。
- \x{hexadecimal #}：十六进制转义。

另外，单引号中可以通过“\”转义符输出的特殊字符只有：

- \'：转义为单引号本身，而不作为字符串标识符。
- \\：用于在单引号前的反斜杠转义为其本身。

下面通过实例来讲解它们的不同用法。

【例 5.1】（实例文件：ch05\5.1.php）

```
<?php
  $message = "PHP 程序";                          //定义字符串变量
  echo "这是关于字符串的程序。<br />";              //输出字符串变量
  echo "这是一个关于双引号和\$的$message<br />";   //使用转义字符
  $message2 = '字符串的程序。';                     //使用单引号赋值字符串变量
  echo '这是一个关于字符串的程序。<br /> ';          //输出字符串变量
  echo '这是一个关于单引号的$message2';
  echo $message2;
?>
```

运行结果如图 5-1 所示。可见单引号串和双引号串在 PHP 的中处理普通的字符串时效果是一样的，而在处理变量时是不一样的，单引号串中的内容只是被当成普通的字符串处理，而双引号串的内容是可以被解释并替换的。

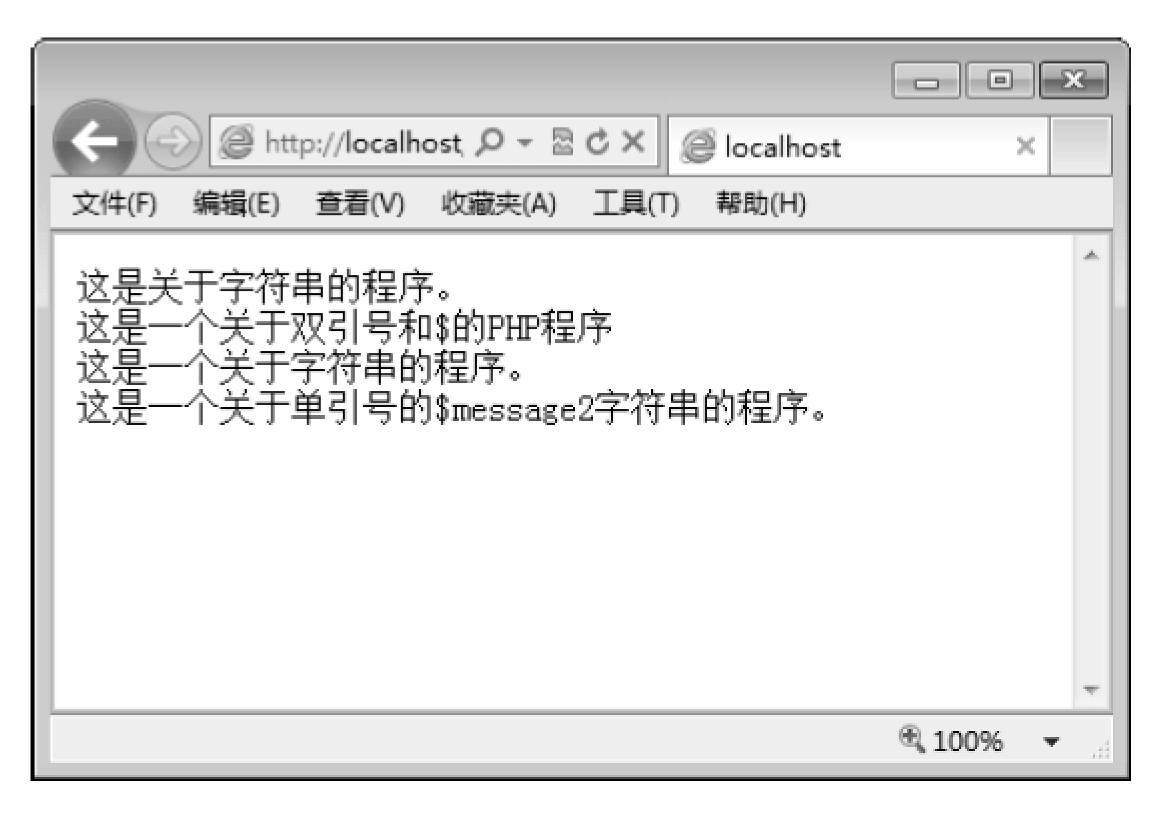


图 5-1　程序运行结果

【案例分析】：

（1）其中，第一段程序使用双引号对字符串进行处理，“\$”转义成了美元符号，$message 的值"PHP 程序"被输出。

（2）第二段程序使用单引号对字符串进行处理。$message2 的值在单引号的字符串中无法被输出，但是可以通过变量被直接打印出来。

5.2　字符串的连接符

字符串连接符的使用十分常见。这个连接符就是“.”（点），它可以直接连接两个字符串，可以连接两个字符串变量，也可以连接字符串和字符串变量，如下面的实例所示。

【例 5.2】（实例文件：ch05\5.2.php）

```
<?php
  //定义字符串
  $a ="使用字符串的连接符";
```

```
  $b= "可以非常方便地连接字符串";
 //连接上面两个字符串 中间用逗号分隔
  $c = $a.", ".$b;    //输出连接后的字符串
  echo $c;
?>
```

运行结果如图 5-2 所示。

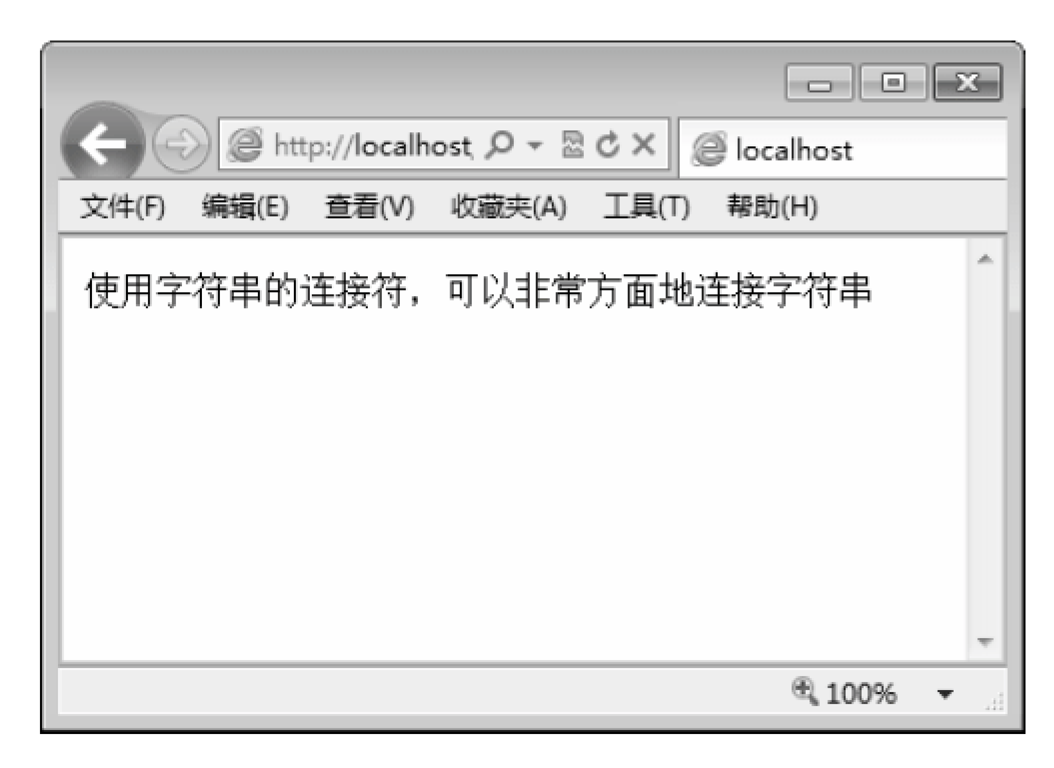


图 5-2　程序运行结果

除了上面的方法以外，读者还可以使用{}的方法连接字符串，此方法类似于 C 语言中 printf 的占位符。下面举例说明其使用方法。

【例 5.3】（实例文件：ch05\5.3.php）

```
<?php
  //定义需要插入的字符串
  $a ="张先生";
  //生成新的字符串
   $b= "欢迎{$a}入住丰乐园高级酒店";
  //输出连接后的字符串
  echo $b;
?>
```

运行结果如图 5-3 所示。



图 5-3　程序运行结果

5.3 字符串的基本操作

字符串的基本操作主要包括对字符串的格式化处理、连接切分字符串、查找字符串、替换字符串等。

5.3.1 手动和自动转义字符串中的字符

手动转义字符串数据，就是在引号内（包括单引号和双引号）通过“\”反斜杠使一些特殊字符转义为普通字符。在介绍单引号和双引号的时候已经对这个方法进行了详细的描述。

自动转义字符串的字符，是通过 PHP 的内置函数 addslashes()来完成的。还原这个操作则是通过 stripslashes()来完成的。以上两个函数，也经常使用在格式化字符串中以用于 MySQL 的数据库存储。

5.3.2 计算字符串的长度

计算字符串的长度经常在很多应用中出现，比如输入框输入文字的长度等，都会用到此功能。使用 strlen()函数就可以实现这个功能。以下实例介绍计算字符串长度的方法和技巧。

【例 5.4】（实例文件：ch05\5.4.php）

```
<?php
  $someinput = "这个字符串的长度不长。length is not long."; //定义字符串变量
  $length = strlen($someinput);                          //获取字符串变量的长度
  if(strlen($someinput)>50){
      echo "输入的字符串的长度不能大于50个字符。";
  }else{
      echo "允许输入字符串的长度，此字符串长度为$length";
  }
?>
```

运行结果如图 5-4 所示。



图 5-4 程序运行结果

【案例分析】：

（1）其中$someinput 为一个字符串变量。strlen($someinput)则直接调用 strlen()函数计算出字符串的长度。

（2）在 if 语句中 strlen($someinput)返回字符串长度并与 50 这一上限作比较。由于$someinput 中有中文和英文两种字符，它的长度为 41，正如输出所示。

（3）由于每个中文字占两个字符位，每个英文字符占一个字符位，字符串内的每个空格也算一个字符位，所以，最后字符串的长度为 41 个字符。

5.3.3 字符串单词统计

有时对字符串的单词进行统计有更大意义。使用 str_word_count()函数可以实现此操作，但是这个函数只对基于 ASCII 码的英文单词起作用，并不对 UTF8 的中文字符起作用。

下面通过实例介绍字符串单词统计中的应用和技巧。

【例 5.5】（实例文件：ch05\7.php）

```
<?php
    $someinput = "How many words in this sentance? Just count it.";    //定义
字符串变量
    $someinput2 = "这个句子由多少个汉字组成？数一数也不知道。";
    echo str_word_count($someinput)."<br />";                  //计算单词个数
    echo str_word_count($someinput2);
?>
```

运行结果如图 5-5 所示。可见 str_word_count()函数无法计算中文字符，查询结果为 0。



图 5-5 程序运行结果

5.3.4 清理字符串中的空格

空格在很多情况下是不必要的，所以清除字符串中的空格显得十分重要。比如在判定输入是否正确的程序中，出现了不必要的空格，将增大程序出现错误判断的机率。

清除空格要用到 ltrim()、rtrim()和 trim()函数。其中 ltrim()是从左面清除字符串头部的空格。rtrim()是从右面清除字符串尾部的空格。trim()则是从字符串两边同时去除头部和尾部的空格。

以下实例介绍去除字符串中空格的方法和技巧。

【例 5.6】（实例文件：ch05\5.6.php）

```
<?php
    $someinput = " 这个字符串的空格有待处理。 ";        //定义字符串变量
    echo "Output:".ltrim($someinput)."End <br />";      //清理字符串头部的空格
    echo "Output:".rtrim($someinput)."End <br />";      //清理字符串头部尾部的空格
    echo "Output:".trim($someinput)."End <br />";       //同时去除头部和尾部的空格
    $someinput2 = " 这个字符串 的 空格有待处理。 ";    //定义中间有空格的字符串变量
```

```
    echo "Output:".trim($someinput2)."End";            //同时去除头部和尾部的空格
?>
```

运行结果如图 5-6 所示。

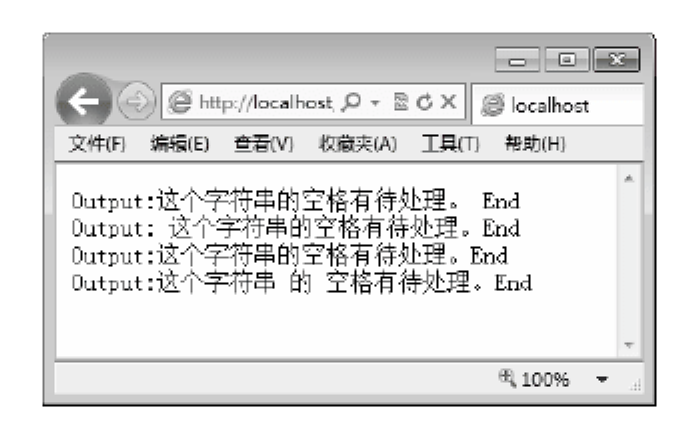


图 5-6　程序运行结果

【案例分析】：

（1）其中，$someinput 为一个两端都有空格的字符串变量。ltrim($someinput)从左边去除空格，rtrim($someinput)从右面去除空格，trim($someinput)从两边同时去除，得到输出如图 5-6 所示。

（2）其中$someinput2 为一个两端都有空格，并且中间也有空格的字符串变量。用 trim($someinput2)处理，但还只是去除两边的空格。

5.3.5　字符串的切分与组合

字符串的切分使用 explode()和 strtok()函数。切分的反向操作为组合，使用 implode()和 join()函数。其中 explode()把字符串切分成不同部分后，存入一个数组。Impolde()函数则是把数组中的元素按照一定的间隔标准组合成一个字符串。

以下实例介绍去除字符串切分和组合的方法和技巧。

【例 5.7】（实例文件：ch05\5.7.php）

```
<?php
    $someinput = "How_to_split_this_sentance.";      //定义字符串变量
    $someinput2 = "把 这个句子 按空格 拆分。";  //定义按空格拆分的字符串
    $a = explode('_',$someinput);                   //切分字符串 someinput
    print_r($a);                              //输出切分后的字符串
    $b = explode(' ',$someinput2);
    print_r($b);
    echo implode('>',$a)."<br />";                 //组合字符串$a
    echo implode('*',$b);
?>
```

运行结果如图 5-7 所示。

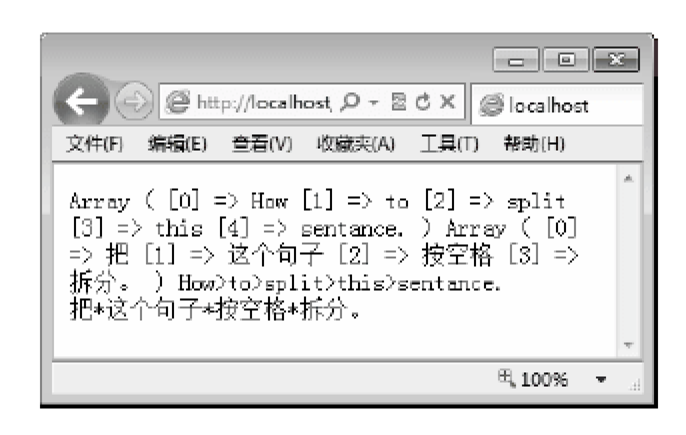


图 5-7　程序运行结果

【案例分析】：

（1）其中 explode()函数把$someinput 和$someinput2 按照下划线和空格的位置把它们分别切分成$a 和$b 两个数组。

（2）其中 implode()函数把$a 和$b 两个数组的元素分别按照“>”和“*”为间隔组合成新的字符串。

5.3.6　字符串子串的截取

在一个字符串中截取一个子串，就是字符串截取。

完成这个操作需要用到 substr()函数。这个函数有 3 个参数，分别规定了目标字符串、起始位置和截取长度。它的格式如下：

```
substr（目标字符串，起始位置，截取长度）
```

其中目标字符串是某个字符串变量的变量名，起始位置和截取长度都是整数。

如果都是正数，起始位置的整数必须小于截取长度的整数，否则函数返回值为假。

如果截取长度为负数，则意味着，是从起始位置开始往后除去从目标字符串结尾算起的长度数的字符以外的所有字符。

以下实例介绍去除字符串截取的方法和技巧。

【例 5.8】（实例文件：ch05\5.8.php）

```
<?php
   $someinput = "create a substring of this string.";   //定义字符串变量$someinput
   $someinput2 = "创建一个这个字符串的子串。";
   echo substr($someinput,0,11)."<br />";          //截取字符串前11个字符
   echo substr($someinput,1,15)."<br />";          //截取从第二个字符开始的前15个字符
   echo substr($someinput,0,-2)."<br />";          //截取除最右侧两个字符外的字符
   echo substr($someinput2,0,12)."<br />";        //截取字符串前12个字符
   echo substr($someinput2,0,10)."<br />";        //截取字符串前10个字符
   echo substr($someinput2,0,11);              //截取字符串前11个字符
?>
```

运行结果如图 5-8 所示。

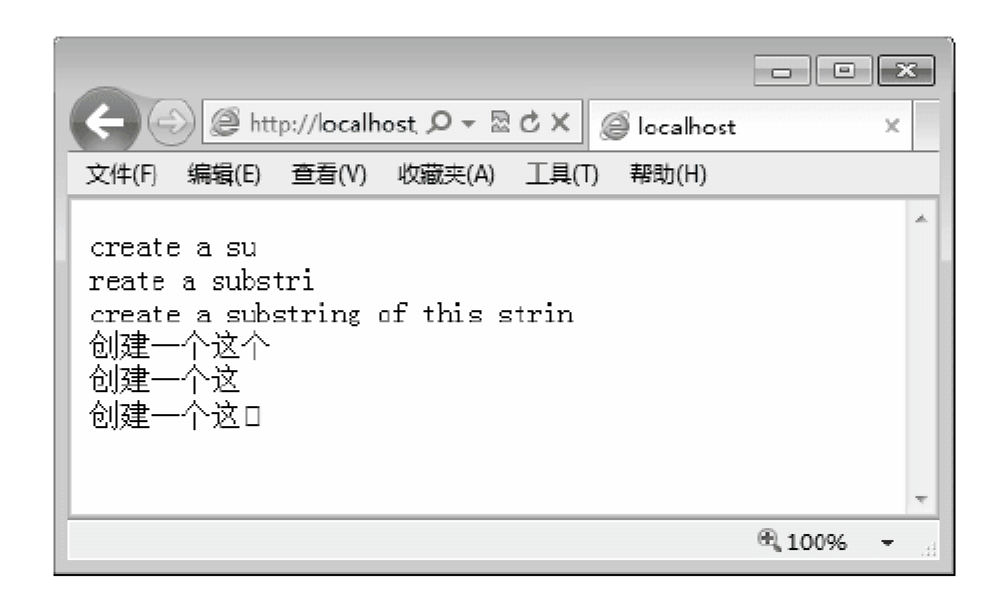


图 5-8　程序运行结果

【案例分析】：

（1）其中$someinput 为英文字符串变量。substr($someinput,0,11)和 substr($someinput,1,15)展示了起始位和截取长度。substr($someinput,0,-2)则是从字符串开头算起，除了最后两个字符，其他字符都截取的子字符串。

（2）其中$someinput2 为中文字符串变量。因为中文字符是全角字符，都占两个字符位，所以截取长度一定要是偶数。如果是单数则在此字符位上的汉字将不被输出。如果在这样截取长度为单数的字符串子串后连接其他字符串输出，会出现输出错误。所以，要小心使用。

5.3.7 字符串子串替换

在某个字符串中替换其中的某个部分是重要的应用，就像在使用文本编辑器中的替换功能一样。

完成这个操作需要使用 substr_replace()函数，它的格式为：

```
substr_replace(目标字符串，替换字符串，起始位置，替换长度)
```

以下实例介绍字符串替换的方法和技巧。

【例 5.9】（实例文件：ch05\5.9.php）

```
<?php
    $someinput = "ID:125846843388648";                    //定义字符串变量
    echo substr_replace($someinput,"************",3,11)."<br />"; //字符串字串替换
    echo substr_replace($someinput,"尾号为",3,11);              //输出替换后的字符串
?>
```

运行结果如图 5-9 所示。

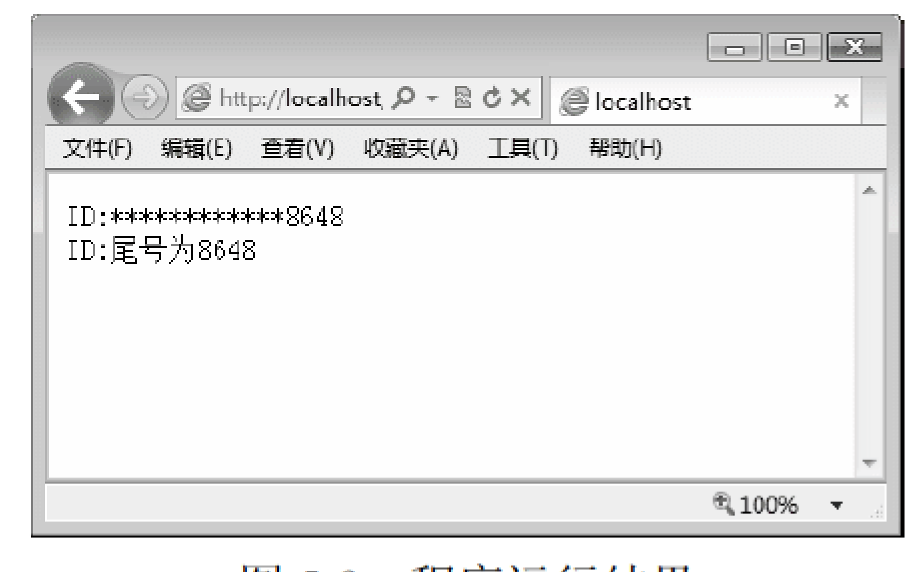


图 5-9　程序运行结果

【案例分析】：

（1）其中，$someinput 为英文字符串变量。从第三个字符开始为 ID 号。第一个输出是以“************”替换第三个字符开始往后的 11 个字符。

（2）第二个输出是用“尾号为”替代第三个字符开始往后的 11 个字符。

5.3.8 字符串查找

在一个字符串中查找另外一个字符串，就像文本编辑器中的查找一样。实现这个操作需要用到 strstr()或 stristr()函数。其格式为：

```
strstr（目标字符串，需查找字符串）
```

如果函数找到需要查找的字符或字符串，则返回从第一个查找到字符串的位置往后所有的字符串内容。

stristr()函数为不敏感查找，也就是对字符的大小写不敏感。用法与 strstr()相同。

以下实例介绍字符串查找的方法和技巧。

【例 5.10】（实例文件：ch05\5.10.php）

```
<?php
   $someinput = "I have a Dream that to find a string with a dream."; //定义英文字符串
   $someinput2 = "我有一个梦想，能够找到理想。";          //定义中文字符串
   echo strstr($someinput,"dream")."<br />";                 //查找指定的字符串
   echo stristr($someinput,"dream")."<br />";
   echo strstr($someinput,"that")."<br />";
   echo strstr($someinput2,"梦想")."<br />";
?>
```

运行结果如图 5-10 所示。



图 5-10　程序运行结果

【案例分析】：

（1）其中$someinput 为英文字符串变量。strstr($someinput,"dream")大小写敏感，所以输出字符串最后的字符。stristr($someinput,"dream")大小写不敏感，所以直接在第一个大写的匹配字符就开始输出。

（2）其中$someinput2 为中文字符串变量。strstr()函数同样对中文字符起作用。

5.3.9　大小写转换

在 PHP 中，通过使用大小写转换函数，可以修改字符串中字母大小不规范的问题。常见的大小写转换函数如下：

```
srting strtolower (srting str) ;  //转换为小写
```

```
srting strtoupper (srting str) ;  //转换为大写
srting ucfirst (srting str) ;    //整个字符串首字母大写
srting ucwords (srting str) ;   //整个字符串中以空格为分隔符的单词首字母大写
```

【例 5.11】（实例文件：ch05\5.11.php）

```
<?php
 $str = "hello I have a dream" ;        //定义英文字符串
echo strtolower($str)."<br />";       //转换为小写
echo strtoupper($str)."<br />";       //转换为大写
echo ucfirst($str)."<br />";         //整个字符串首字母大写
echo ucwords($str)."<br />";        //整个字符串中以空格为分隔符的单词首字母大写
echo $aa;                  //输出原字符串
 ?>
```

运行结果如图 5-11 所示。

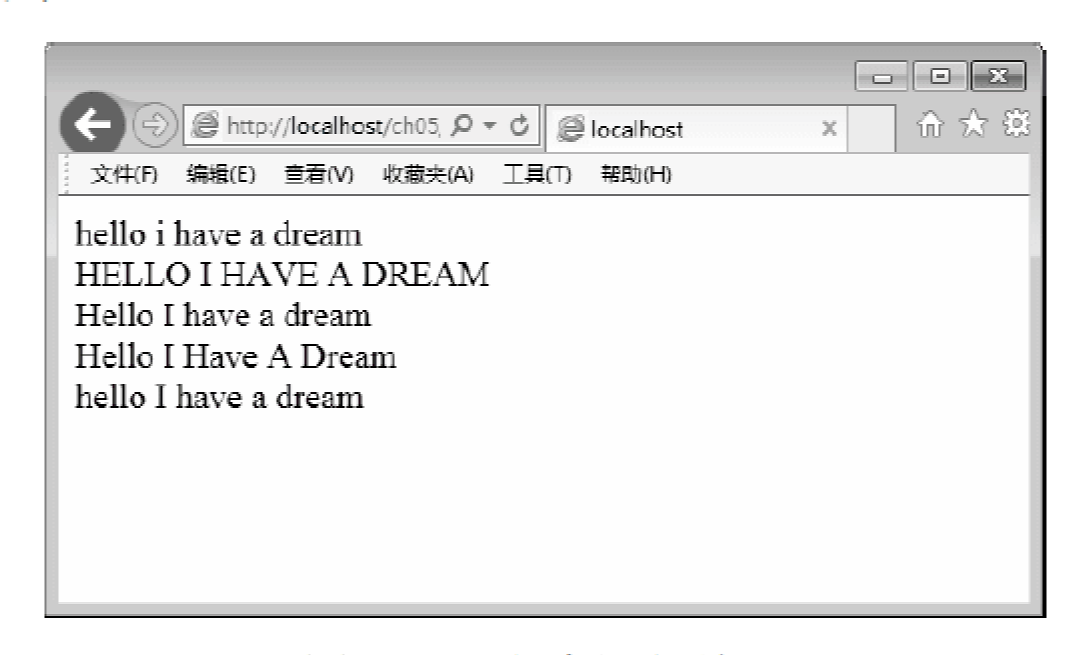


图 5-11 程序运行结果

5.4 什么是正则表达式

上面介绍的对字符串的处理方法比较简单，只是使用一定的函数对字符串进行处理，无法满足对字符串的复杂处理需求，此时就需要使用正则表达式。

正则表达式是把文本或字符串按照一定的规范或模型表示的方法，经常用于文本的匹配操作。

例如，验证用户在线输入的邮件地址的格式是否正确。常常使用正则表达式技术，用户所填写的表单信息将会被正常处理；反之，如果用户输入的邮件地址与正则表达的模式不匹配，将会弹出提示信息，要求用户重新输入正确的邮件地址。可见正则表达式在 Web 应用的逻辑判断中具有举足轻重的作用。

5.5 正则表达式语法规则

一般情况下，正则表达式由两部分组成，分别是元字符和文本字符。元字符就是具有特殊含义的字符，例如“？”和“*”等，文本字符就是普通的文本，例如字母和数字等。本节主要讲述正则表达式的语法规则。

1. 方括号（[ ]）

方括号内的一串字符是将要用来进行匹配的字符。例如正则表达式在方括号内的[name]是指在目标字符串中寻找字母 n、a、m、e。[jjk]表示在目标字符串中寻找字符 j 和 k。

2. 连字符（-）

在很多情况下，不可能逐个列出所有字符。比如，若要匹配所有英文字符，则把 26 个英文字母全部输入会十分困难。这样就有如下表示：

- [a-z]表示匹配英文小写从 a 到 z 的任意字符。
- [A-Z]表示匹配英文大写从 A 到 Z 的任意字符。
- [A-Za-z] 表示匹配英文大小写从大写 A 到小写 z 的任意字符。
- [0-9]表示匹配从 0 到 9 的任意十进制数。

由于字母和数字的区间固定，所以根据这样的表示方法，即[开始-结束]，程序员可以重新定义区间大小，如[2-7]、[c-f]等。

3. 点号字符（.）

点号字符在正则表达式中是一个通配符，它代表所有字符和数字，例如，“.er”表示所有以 er 结尾的三个字符的字符串，可以是 per、ser、ter、@er、&er 等。

4. 限定符（+*？{n,m}）

加号“+”表示其前面的字符至少有一个。例如，“9+”表示目标字符串包含至少一个 9。

星号“*”表示其前面的字符不止一个或零。例如，“y*”表示目标字符串包含 0 或不止一个 y。

问号“？”表示其前面的字符为一个或零。例如，“y？”表示目标字符串包含 0 或一个 y。

大括号“{n,m}”表示其前面的字符有 n 或 m 个。例如，“a{3，5}”表示目标字符串包含 3 个或 5 个 a。“a{3}”表示目标字符串包含 3 个 a。“a{3，}”表示目标字符串包含至少 3 个 a。

点号和星号可以一起使用，如“.*”表示匹配任意字符。

5. 行定位符（^和$）

行定位符用来确定匹配字符串所要出现的位置。

如果是在目标字符串开头出现，则使用符号“^”；如果是在目标字符串结尾出现，则使用符号“$”。例如，^xiaoming 是指 xiaoming 只能出现在目标字符串开头。8895$ 是指 8895 只能出现在目标字符串结尾。

同时使用“^$”这两个符号，如“^[a-z]$”，表示目标字符串要只包含从 a 到 z 的单个字符。

6. 排除字符（[^]）

符号“^”在方括号内所代表的意义则完全不同，它表示一个逻辑“否”。排除匹配字符串在目标字符串中出现的可能，例如[^0-9]表示目标字符串包含从 0 到 9“以外”的任意其他字符。

7. 括号字符（( )）

括号字符表示子串，所有对包含在子串内字符的操作，都是以子串为整体进行的。括号字符也是把正则表达式分成不同部分的操作符。

8. 选择字符（|）

选择字符表示“或”选择。例如，“com|cn|com.cn|net”表示目标字符串包含 com 或 cn 或 com.cn 或 net。

9. 转义字符（\）与反斜线（\）

由于“\”在正则表达式中属于特殊字符，如果单独使用此字符，则直接表示为作为特殊字符的转义字符。如果要表示反斜杠字符本身，则在此字符前添加转义字符“\”，即“\\”。

10. 认证 email 的正则表达

在处理表单数据的时候，对用户的 email 进行认证是十分常用的。如何判断用户输入的是一个 email 地址呢？就是用正则表达式匹配。它的格式如下。

```
^[A-Za-z0-9_.]+@[ A-Za-z0-9_]+\.[ A-Za-z0-9.]+$
```

其中^[A-Za-z0-9_.]+表示，至少有一个英文大小写字符、数字、下划线、点号，或者这些字符的组合。@表示 email 中的“@”。[A-Za-z0-9_]+表示，至少有一个英文大小写字符、数字、下划线，或者这些字符的组合。\.表示 email 中“.com”之类的点。由于这里点号只是点本身，所以用反斜杠对它进行转义。[A-Za-z0-9.]+$表示，至少有一个英文大小写字符、数字、点号，或者这些字符的组合，并且直到这个字符串的末尾。

11. 如何使用正则表达式对字符串进行匹配

使用正则表达式对目标字符串进行匹配是正则表达式的主要功能。

完成这个操作需要用到 ereg()函数。这个函数用于在目标字符串中寻找符合特定正则表达规范的字符串子串，根据指定的模式来匹配文件名或字符串。其中 ereg()函数对字符大小写不敏感。它的语法格式如下：

```
ereg（正则表达规范，目标字符串，数组）
```

提示：另外用户也可以使用 eregi()函数对字符串进行匹配，它和 ereg()函数最大区别是对字符大小写敏感。

下面介绍利用正则表达规范匹配 email 输入的方法和技巧。

【例 5.12】（实例文件：ch05\5.12.php）

```
<?php
  $email = "wangxioaming2011@hotmail.com";                //定义字符串
  $email2 = "The email is liuxiaoshuai_2011@hotmail.com";
  $asemail = "This is wangxioaming2011@hotmail";
  $regex = '^[a-zA-Z0-9_.]+@[a-zA-Z0-9_]+\.[a-zA-Z0-9.]+$';  //定义正则表达式
规范
  $regex2 = '[a-zA-Z0-9_.]+@[a-zA-Z0-9_]+\.[a-zA-Z0-9.]+$';
  if(ereg($regex, $email, $a)){                        //利用正则表达式规范字符串
     echo "This is an email.";
   print_r($a);
   echo "<br />";
```

```
    }
    if(ereg($regex2, $email2, $b)){
        echo "This is a new email.";
     print_r($b);
     echo "<br />";
    }
    if(ereg($regex, $asemail)){
        echo "This is an email.";
    }else{
        echo "This is not an email.";
    }
?>
```

运行结果如图 5-12 所示。

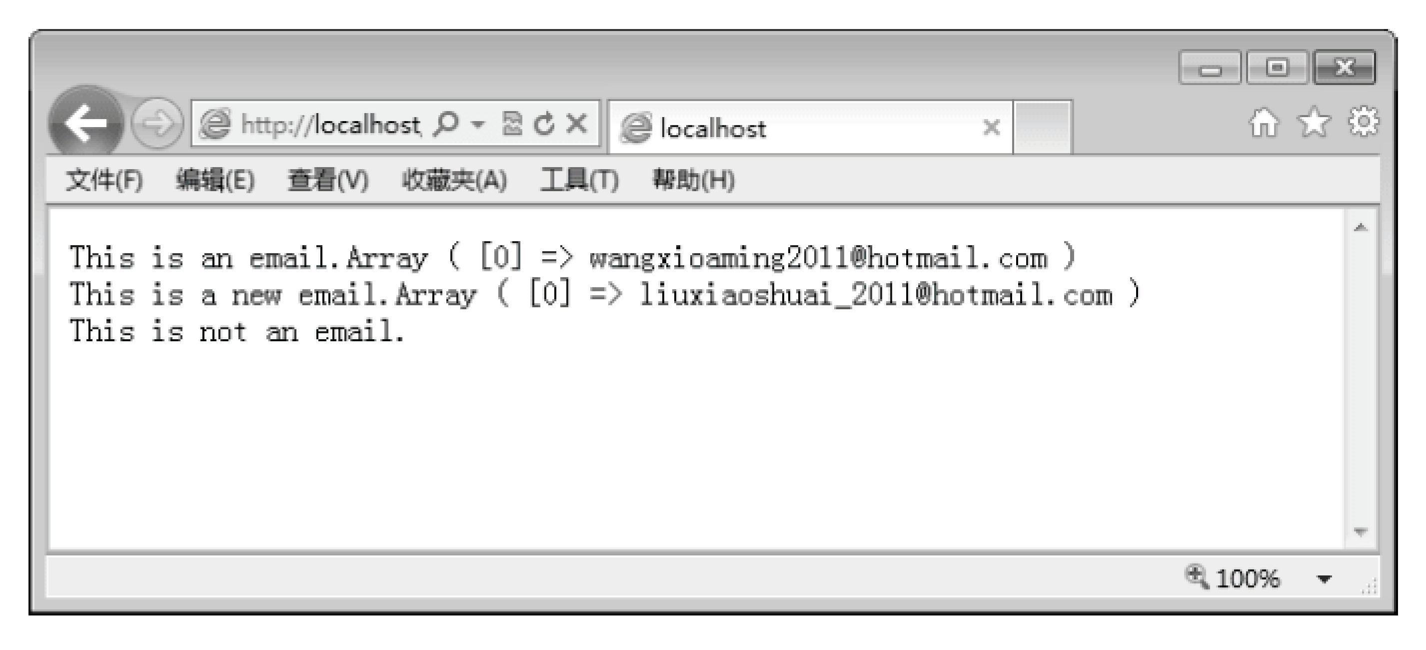


图 5-12　程序运行结果

【案例分析】：

（1）其中，$email 就是一个完整的 email 字符串，用$regex 这个正则规范，也就是匹配 email 的规范来匹配$email，得出的结果为图中第一行输出。

（2）由于 ereg()函数的格式，ereg($regex, $email, $a)把匹配的子串存储在名为$a 的数组中。print_r($a)打印数组，得出结果为第一行数组输出。

（3）其中，$email2 就是一个包含了完整 email 的字符串。用$regex 匹配，其返回值必然为 false。用$regex2 规范匹配，其返回值为真。因为$regex2 规范中去掉了表示从字符串头部开始的符号“^”。ereg($regex2, $email2, $b)把匹配的子串存储在数组$b 中。print_r($b)，得到第二行数组的输出。

（4）$asemail 字符串不符合规范$regex，返回值为 false，得到相应输出。

12. 使用正则表达式替换字符串子串

完成字符串及其子串的匹配后，如果需要对字符串的子串进行替换，也可以使用正则表达式完成。这种需求，比如把输入文本中的 url 完成可以直接点击的连接。此操作需要使用 ereg_replace()和 eregi_replace()函数。其中 ereg_replace()对大小写敏感，而 eregi_replace()对大小写不敏感。其格式为：

```
ereg_replace(正则表达规范，欲取代字符串子串，目标字符串)
```

提示：另外用户也可以使用 eregi_replace ()函数对字符串进行替换，它和 ereg_replace ()函数最

大区别是对字符大小写敏感。

通过以下实例，介绍利用正则表达式取代字符串子串的方法和技巧。

【例 5.13】（实例文件：ch05\5.13.php）

```
<?php
$searchurl = "这是搜索引擎连接：http://www.google.com/和http://www.baidu.com/。";
echo ereg_replace("(http://)([a-zA-Z0-9./-_]+)","<a href=\"\\0\">\\0</a>",$searchurl);
echo "<br />";
echo ereg_replace("(http://)([a-zA-Z0-9./-_]+)","<a href=\"\\0\">\\2</a>",$searchurl);
?>
```

运行结果如图 5-13 所示。

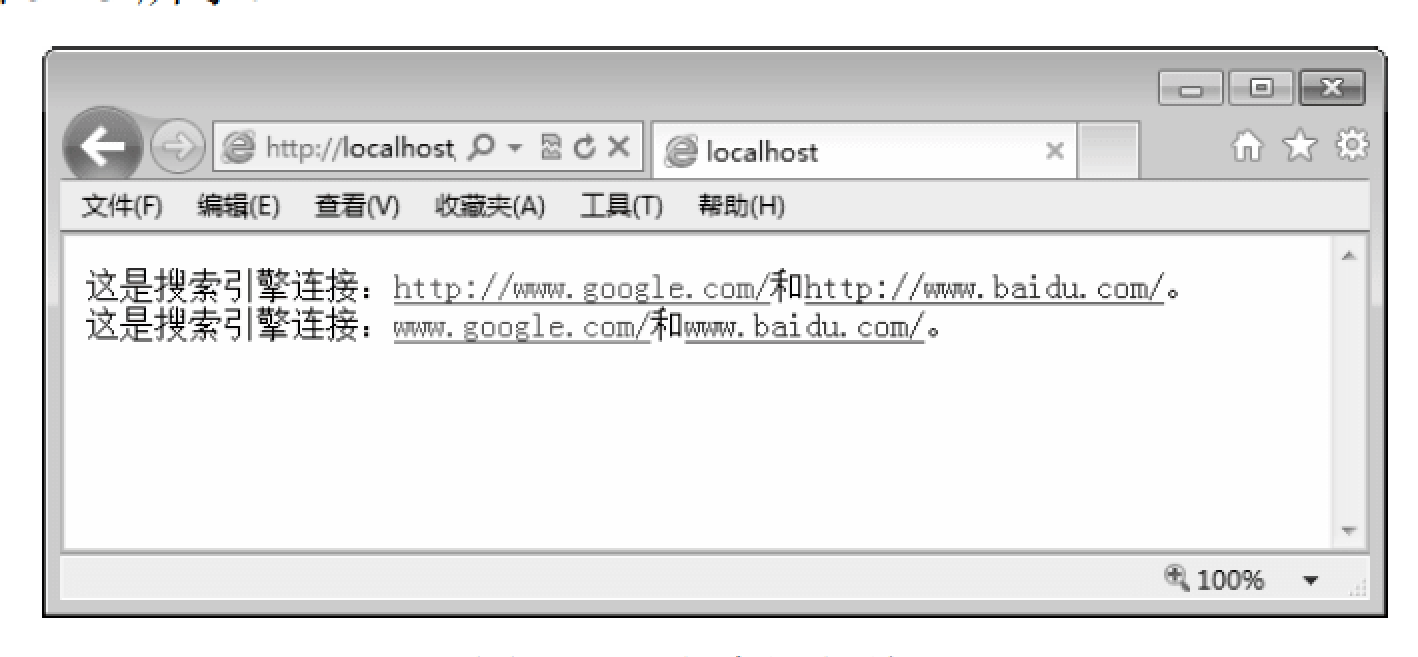


图 5-13　程序运行结果

【案例分析】：

（1）其中，$searchurl 里面包含两个 url 文本。ereg_replace()按照格式对$searchurl 里的 url 进行匹配替换。

（2）正则规范为"(http://)([a-zA-Z0-9./-_]+)"，它分为两部分，即(http://)和([a-zA-Z0-9./-_]+)，前者直接匹配，后者用正则语法匹配。

（3）第一行的输出，替换为"\\0"。里面的“\\0”把反斜杠转义后表示的是“\0”，“\0”表示正则规则中所有部分匹配的内容。第二行的输出，替换为"\\2"，里面的“\\2”把反斜杠转义后表示的是“\2”，“\2”表示正则规则中第二部分匹配的内容，输出如图 5-12 所示。依次类推，“\1”表示的是第一部分匹配的内容，即（http://）。

13. 使用正则表达式切分字符串

使用正则表达式可以把目标字符串按照一定的正则规范切分成不同的子串。完成此操作要用到 strtok()函数，它的语法格式为：

```
strtok（正则表达式规范，目标字符串）
```

这个函数指以正则规范内出现的字符为准，把目标字符串切分成若干个子串，并且存入数组。

下面通过实例，介绍利用正则表达式切分字符串的方法和技巧。

【例 5.14】（实例文件：ch05\5.14.php）

```
<?php
  $string = "Hello world. Beautiful day today."; //定义字符串
  $token = strtok($string, " ");            //切分字符串
  while ($token !== false)              //使用 while 循环输出切分后的字符串
    {
    echo "$token<br />";
    $token = strtok(" ");
    }
?>
```

运行结果如图 5-14 所示。



图 5-14　程序运行结果

【案例分析】：

（1）其中，$string 为包含多种字符的字符串。strtok($string, " ")对其进行切分，并将结果存入数组$token。

（2）其正则规范为"　"，是指以空格将字符串切分。

5.6　实战演练——创建酒店系统在线订房表

本实例主要创建酒店系统的在线订房表，其中需要创建两个 PHP 文件，具体创建步骤如下。

01 在网站主目录下建立文件 formstringhandler.php，输入以下代码并保存。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/ DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<HEAD><meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
您的订房信息：</HEAD>
<BODY>
<?php
 $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
 $customername = trim($_POST['customername']);
 $gender = $_POST['gender'];
 $arrivaltime = $_POST['arrivaltime'];
 $phone = trim($_POST['phone']);
 $email = trim($_POST['email']);
 $info = trim($_POST['info']);
```

```
if(!eregi('^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9_\-\.]+$',$email)){
        echo "这不是一个有效的 email 地址，请返回上页且重试";
        exit;
    }
    if(!eregi('^[0-9]$',$phone) and strlen($phone)<= 4 or strlen($phone)>= 15){
        echo "这不是一个有效的电话号码，请返回上页且重试";
        exit;
    }
    if( $gender == "m"){
       $customer = "先生";
    }else{
       $customer = "女士";
    }
    echo '<p>您的订房信息已经上传，我们正在为您准备房间。 确认您的订房信息如下:</p>';
    echo $customername."\t".$customer.' 将会在 '.$arrivaltime.' 天后到达。 您的电话
为'.$phone."。我们将会发送一封电子邮件到您的 email 邮箱: ".$email."。<br /><br />另外，我
们已经确认了您其他的要求如下: <br /><br />";
    echo nl2br($info);
    echo "<p>您的订房时间为:".date('Y m d H: i: s')."</p>";
   ?>
   </BODY>
   </HTML>
```

02 在网站主目录下创建文件 form4string.html，输入以下代码并保存。

```
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/ DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
   <HEAD><meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/><h2>GoodHome 在线订房表。</h2></HEAD>
   <BODY>
   <form action="formstringhandler.php" method="post">
   <table>
   <tr bgcolor="#3399FF" >
       <td>客户姓名:</td>
       <td><input type="text" name="customername" size="20" /></td>
   </tr>
   <tr bgcolor="#CCCCCC" >
       <td>客户性别: </td>
       <td>
        <select name="gender">
           <option value="m">男</option>
           <option value="f">女</option>
        </select>
        </td>
   </tr>
   <tr bgcolor="#3399FF" >
       <td>到达时间:</td>
       <td>
```

```
        <select name="arrivaltime">
           <option value="1">一天后</option>
           <option value="2">两天后</option>
           <option value="3">三天后</option>
           <option value="4">四天后</option>
           <option value="5">五天后</option>
        </select>
       </td>
   </tr>
   <tr bgcolor="#CCCCCC" >
       <td>电话:</td>
       <td><input type="text" name="phone" size="20" /></td>
   </tr>
   <tr bgcolor="#3399FF" >
       <td>email:</td>
       <td><input type="text" name="email" size="30" /></td>
   </tr>
   <tr bgcolor="#CCCCCC" >
       <td>其他需求:</td>
       <td> <textarea name="info" rows="10" cols="30">    如果您有什么其他要求，请填在这里。</textarea>
       </td>
   </tr>
   <tr bgcolor="#666666" >
       <td align="center"><input type="submit" value="确认订房信息" /></td>
   </tr>
   </table>
   </form>
   </BODY>
   </HTML>
```

03 运行 form4string.html，结果如图 5-15 所示。



图 5-15 程序运行结果

04 填写表单。【客户姓名】为“王小明”、【客户性别】为“男”、【到达时间】为“三天后”、【电话】为 13592XXXX77、【email】为 wangxiaoming@hotmail.com、【其他需求】为“两壶开水，【Enter】一条白毛巾，【Enter】一个冰激凌”。单击【确认订房信息】按钮，浏览器会自动跳转至 formstringhandler.php 页面，显示结果如图 5-16 所示。

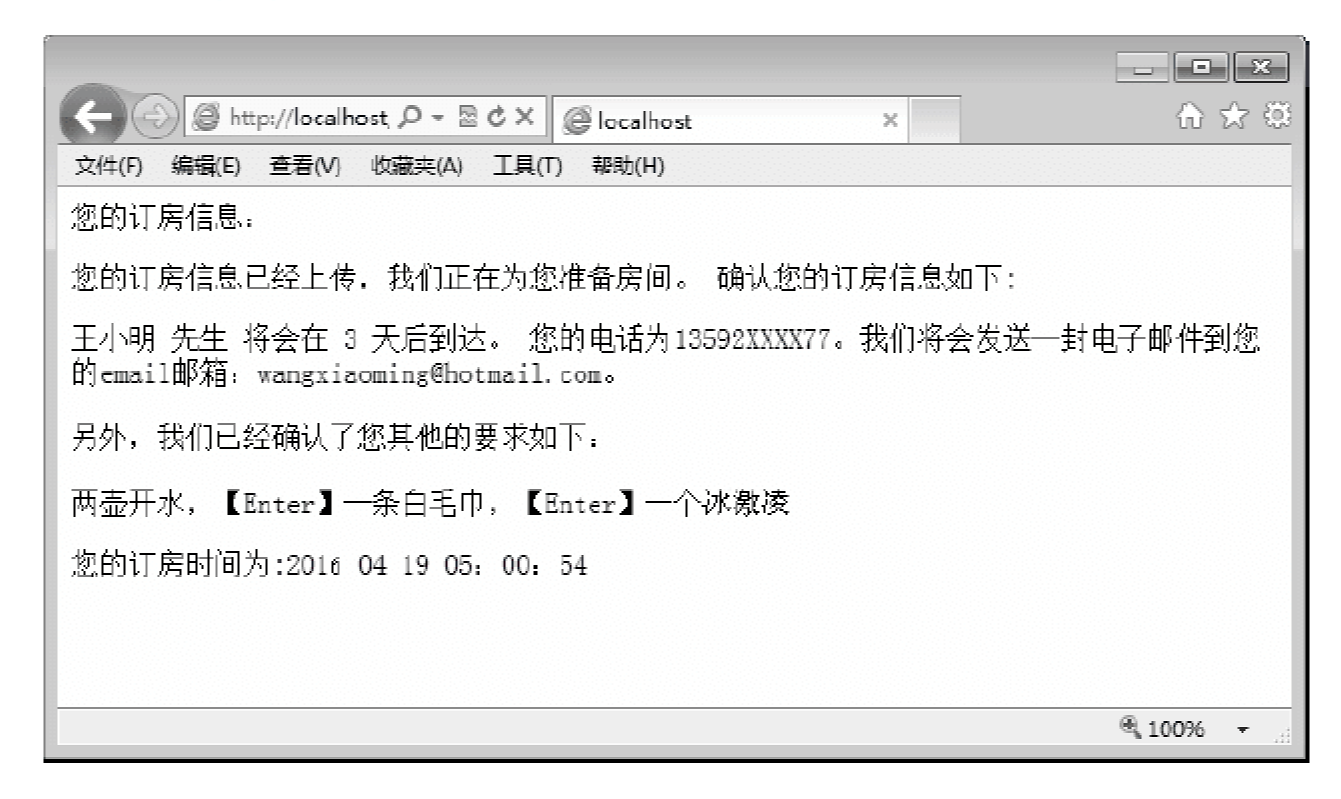


图 5-16　程序运行结果

【案例分析】：

（1）$customername = trim($_POST['customername']);、$phone = trim($_POST['phone']);、$email = trim($_POST['email']); 和$info = trim($_POST['info']); 都是通过文本输入框直接输入的。所以，为了保证输入字符串的纯粹性，以方便处理，则需要使用 trim()来对字符串前后的空格进行清除。另外，ltrim()清除左边的空格；rtrim()清除右边的空格。

（2）!eregi('^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9_\-\.]+$',$email)中使用正则表达式对输入的 email 文本进行判断。

（3）nl2br()对$info 变量中的【Enter】操作，也就是对
操作符进行了处理。在有新行“\nl”操作的地方生成
。

（4）其中，由于要显示中文，需要对文字编码进行设置，charset=gb2312 就是简体中文的文字编码。

5.7　高手私房菜

技巧 1：模式修饰符、单词界定符和方括号“[]”连用，而是和“/”在一起使用？

在 PHP 正则表达式的语法当中，一种是 POSIX 语法，一种是 Perl 语法。POSIX 语法是先前所介绍的语法。Perl 语法则不同于 POSIX 语法。Perl 语法的正则表达是以“/”开头和以“/”结尾的，如“/name/”便是一个 Perl 语法形式的正则表达。

模式修饰符则是在 Perl 语法正则表示中的内容。比如“i”表示正则表达式对大小写不敏感，“g”表示找到所有匹配字符，“m”表示把目标字符串作为多行字符串进行处理，“s”表示把目标字符串作为单行字符串进行处理，忽略其中的换行符，“x”表示忽略正则表达式中的空格和备注，“u”表示在首次匹配后停止。

单词界定符也是 Perl 语法正则表示中的内容。不同的单词界定符表示不同的字符界定范围。

比如以下单词界定符的表示意义为：

“\A”表示仅仅匹配字符串的开头。“\b”表示匹配到单词边界。“\B”表示除了单词边界，匹配所有。“\d”表示匹配所有数字字符，等同于“[0-9]”。“\D”表示匹配所有非数字字符。“\s”表示匹配空格字符。“\S”表示匹配非空格字符。“\w”表示匹配字符串，如同“[a-zA-Z0-9_]”。“\W”表示匹配字符，忽略下划线和字母数字字符。

技巧 2：支持 Perl 语法形式的正则表达式有哪些？

PHP 为 Perl 语法的正则表达方式提供了如下函数：

（1）preg_grep()用来搜索一个数组中的所有数组元素，以得到匹配元素。

（2）preg_match()以特定模式匹配目标字符串。

（3）preg_match_all()以特定模式匹配目标字符串，并且把匹配元素作为元素返回给一个特定数组。

（4）preg_quote()在每一个正则表达式的特殊字符前插入一个反斜杠“\”。

（5）preg_replace()替代所有符合正则表达式格式的字符，并返回按照要求修改的结果。

（6）preg_replace_callback()以键值代替所有符合正则表达式格式字符的键名。

（7）preg_split()按照正则模型切分字符串。

5.8 经典习题

（1）制作一个包含单引号和双引号的例子，并分析它们的区别。

（2）制作一个包含连接字符串的例子，并输出连接后的结果。

（3）制作一个包含计算字符串长度、替换字符串中的空格、字符串字串截取和替换的例子。

（4）制作一个包含正则表达式的例子。

（5）制作一个酒店系统在线订房表的例子。

第 6 章　PHP 数组

数组在 PHP 中是极为重要的数据类型。本章将介绍什么是数组，数组的类型、数组的构造、遍历数组、数组排序、向数组中添加和删除元素、查询数组中的指定元素、统计数组的元素个数、删除数组中重复的元素、数组的序列化等操作。通过本章的学习，读者可以掌握数组的常用操作和技巧。

本章学习目标

- 了解什么是数组
- 熟悉数组的类型
- 掌握数组构建的方法
- 掌握遍历数组的方法
- 掌握数组排序的方法
- 掌握数组和字符串之间转换的方法
- 掌握向数组中添加和删除元素的方法
- 掌握查询数组中指定元素的方法
- 掌握统计元素个数的方法
- 掌握删除数组中重复元素的方法
- 掌握调换数组中的键值和元素值的方法
- 掌握如何实现数组的序列化

6.1　什么是数组

什么是数组？数组，就是用来存储一系列数值的地方。数组（array）是非常重要的数据类型。相对于其他的数据类型，它更像是一种结构，而这种结构可以存储一系列的数值。

数组中的数值被称为数组元素（element）。每一个元素都有一个对应的标识（index），也称作键值（key）。通过这个标识，可以访问数组元素。数组的标识可以是数字也可以是字符串。

例如一个班级通常有十几个人，如果要找出某个学生，可以利用学号来区分每一个队员，这时，班级就是一个数组，而学号就是下标，如果指明学号，就可以找到对应的学生。

6.2　数组的类型

数组分为数字索引数组和关联索引数组。本节将详细讲述这两种数组的使用方法。

6.2.1　数字索引数组

数字索引数组是最常见的数组类型，默认从 0 开始计数。另外，数组变量在使用时即可创建，创建时即可使用。

声明数组的方法有两种。

（1）使用 array()函数声明数组，具体的声明数组方式如下。

array 数组名称（[mixed]），其中参数 mixed 的语法为 key=>value，如果有多个 mixed，可以用逗号分开，分别定义了索引和值。

```
$arr = array("1"=> "空调", "2"=>"冰箱", "3"=>"洗衣机", "4"=>"电视机");
```

利用 array()函数定义比较方便，可以只给出数组的元素值，而不需要给出键值，例如：

```
$arr = array( "空调","冰箱","洗衣机","电视机");
```

（2）直接通过为数组元素赋值的方式声明数组。

如果在创建数组时不知道数组的大小，或者数组的大小可能会根据实际情况发生变化，此时可以使用直接赋值的方式声明数组，例如：

```
$arr[1]= "空调";
$arr[2]= "冰箱";
$arr[3]= "洗衣机";
$arr[4]= "电视机";
```

下面以酒店网站系统中酒店房价为例进行讲解。

【例 6.1】（实例文件：ch06\6.1.php）

```
<?php
  $roomtypes = array( '单床房','标准间','三床房','VIP 套房');
  echo
$roomtypes[0]."\t".$roomtypes[1]."\t".$roomtypes[2]."\t".$roomtypes[3]."<br
/>";
  echo "$roomtypes[0] $roomtypes[1] $roomtypes[2] $roomtypes[3] <br />";
  $roomtypes[0] = '单人大床房';
  echo "$roomtypes[0] $roomtypes[1] $roomtypes[2] $roomtypes[3]<br />";
?>
```

运行结果如图 6-1 所示。

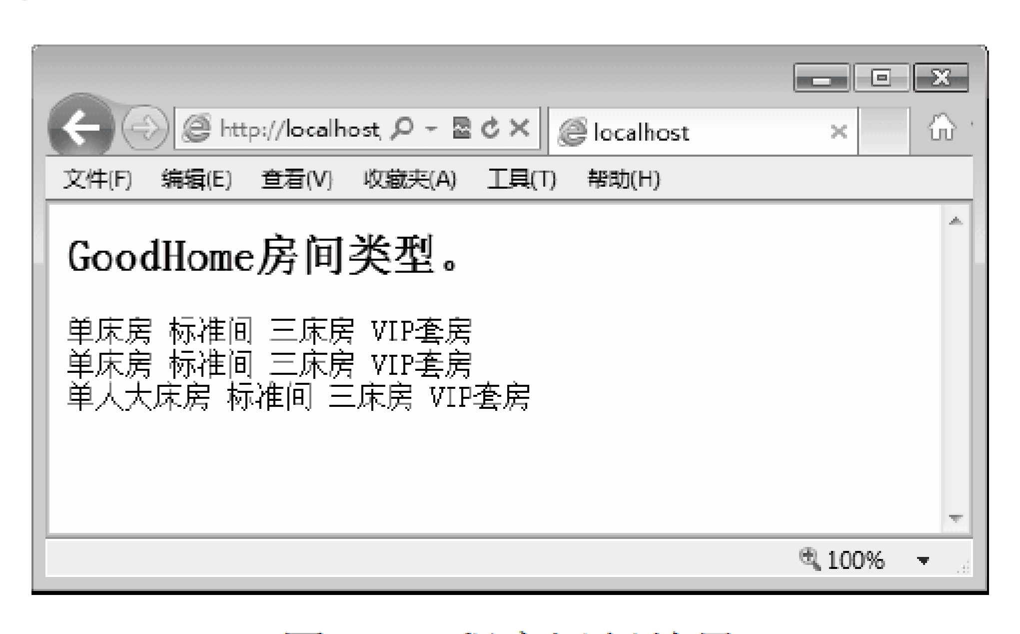


图 6-1　程序运行结果

【案例分析】：

（1）其中，$roomtypes 为一维数组，用关键字 array 声明，并且用“=” 赋值给数组变量

$roomtypes 。

（2）'单床房'、'标准间'、'三床房'和'VIP 套房' 为数组元素，且这些元素为字符串型，用单引号方式表示。每个数组元素用“，” 分开。echo 命令直接打印数组元素，元素索引默认从 0 开始，所以第一个数组元素为$roomtypes[0]。

（3）数组元素可以直接通过“=” 号赋值，如$roomtypes[0] = '单人大床房'; ，echo 打印后为“单人大床房”。

6.2.2 关联索引数组

关联数组的键名可以是数值和字符串混合的形式，而不像数字索引数组的键名只能为数字。所以判断一个数组是否为关联数组的依据是：数组中的键名是否存在不是数字的字符，如果存在，则为关联数组。

下面以使用关联索引数组编写酒店房间类型为例进行讲解。

【例 6.2】（实例文件：ch06\6.2.php）

```
<?php
  $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'VIP 套房'=> 368);
  echo $prices_per_day['标准间']."<br />";
?>
```

运行结果如图 6-2 所示。



图 6-2 程序运行结果

【案例分析】：

其中 echo 命令直接指定数组$prices_per_day 中的关键字索引 standardroom（是一个字符串）便可打印出数组元素 268（是一个整型数字）。

6.3 PHP 常量数组

在 PHP 5.6 中仅能通过 const 定义常量数组，例如以下例子：

```
<?php
// 使用 const 函数来定义数组
const arr = array( "空调","冰箱","洗衣机","电视机");
echo arr[2];
```

```
?>
```

以上程序执行后输出结果为：洗衣机。

PHP 7 可以通过 define() 来定义常量数组。例如以下例子：

```
<?php
// 使用 define 函数来定义数组
define ('学员', [
   '张笑笑',
   '杨洋',
   '王一刀'
]);
print(学员[1]);
?>
```

以上程序执行后输出结果为：杨洋。

6.4 数组构造

按照数组的构造来分，可以把数组分为一维数组和多维数组。

6.4.1 一维数组

数组中每个数组元素都是单个变量，不管是数字索引还是关联索引，这样的数组为一维数组。

【例 6.3】（实例文件：ch06\6.3.php）

```
<?php
  $roomtypes = array( '单床房','标准间','三床房','VIP 套房');
  $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'VIP 套房
'=> 368);
?>
```

其中$roomtypes 和$prices_per_day 都是一维数组。

6.4.2 多维数组

数组也是可以“嵌套”的，即每个数组元素也可以是一个数组，这种含有数组的数组就是多维数组，例如：

```
<?php
  $roomtypes = array( array( 'type'=>'单床房',
                          'info'=>'此房间为单人单间。',
                          'price_per_day'=>298
                        ),
                   array( 'type'=>'标准间',
                           'info'=>'此房间为两床标准配置。',
                          'price_per_day'=>268
                        ),
```

```
                    array( 'type'=>'三床房',
                           'info'=>'此房间备有三张床',
                           'price_per_day'=>198
                         ),
                    array( 'type'=>'VIP 套房',
                           'info'=>'此房间为 VIP 两间内外套房',
                           'price_per_day'=>368
                         )

  );
?>
```

其中$roomtypes 就是多维数组。这个多维数组其实包含两个维数，有点像数据库中的表格，第一个 array 里面的每个数组元素都是一个数组，而这些数组就像数据二维表中的一行记录。这些包含在第一个 array 里面的 array 又都包含 3 个数组元素，分别是 3 个类型的信息，这就像数据二维表中的字段。

可将上面的数组绘制成图，如图 6-3 所示。

	A	B	C	D
1	type	info	price_per_day	
2	单床房	此房间为单人单间。	298	array
3	标准间	此房间为两床标准配置。	268	array
4	三床房	此房间备有三张床	198	array
5	VIP套房	此房间为VIP两间内外套房	368	array
6	ARRAY			

图 6-3　程序运行结果

其实，$roomtypes 就代表了这样的一个数据表。

如果出现了两维以上的数组，比如三维数组，例如：

```
<?php
$buidling = array(array( array( 'type'=>'单床房',
                           'info'=>'此房间为单人单间。',
                           'price_per_day'=>298
                         ),
                    array( 'type'=>'标准间',
                           'info'=>'此房间为两床标准配置。',
                           'price_per_day'=>268
                         ),
                    array( 'type'=>'三床房',
                           'info'=>'此房间备有三张床',
                           'price_per_day'=>198
                         ),
                    array( 'type'=>'VIP 套房',
                           'info'=>'此房间为 VIP 两间内外套房',
                           'price_per_day'=>368
                         )
                  ),
                  array( array( 'type'=>'普通餐厅包房',
```

```
                            'info'=>'此房间为普通餐厅包房。',
                             'roomid'=>201
                         ),
                     array( 'type'=>'多人餐厅包房',
                            'info'=>'此房间为多人餐厅包房。',
                             'roomid'=>206
                         ),
                     array( 'type'=>'豪华餐厅包房',
                            'info'=>'此房间为豪华餐厅包房。',
                             'roomid'=>208
                         ),
                     array( 'type'=>'VIP 餐厅包房',
                            'info'=>'此房间为 VIP 餐厅包房',
                             'roomid'=>310
                         )
                 )
         );
?>
```

这个三维数组，在原来的二维数组后面又增加了一个二维数组，给出了餐厅包房的数据二维表信息。把这两个二维数组作为更外围 array 的两个数组元素，就产生了第三维。这个表述等于用两个二维信息表表示一个名为$building 的数组对象，如图 6-4 所示。

	A	B	C	D	E
1	type	info	price_per_day		
2	单床房	此房间为单人单间。	298	array	
3	标准间	此房间为两床标准配置。	268	array	
4	三床房	此房间备有三张床	198	array	
5	VIP套房	此房间为VIP两间内外套房	368	array	
6	ARRAY（二维）				
7	type	info	roomid		
8	普通餐厅包房	此房间为普通餐厅包房	201	array	
9	多人餐厅包房	此房间为多人餐厅包房。	206	array	
10	豪华餐厅包房	此房间为豪华餐厅包房。	208	array	
11	VIP餐厅包房	此房间为VIP餐厅包房	301	array	ARRAY（三维）
12	ARRAY（二维）				

图 6-4　程序运行结果

6.5　遍历数组

所谓数组的遍历是要把数组中的变量值读取出来。下面讲述遍历数组的常见方法。

6.5.1　遍历一维数字索引数组

下面讲解如何通过循环语句遍历一维数字索引数组。此案例中用到了 for 循环和 foreach 循环。

【例 6.4】（实例文件：ch06\6.4.php）

```
<?php
  $roomtypes = array( '单床房','标准间','三床房','VIP 套房');
  for ($i = 0; $i < 3; $i++){
     echo $roomtypes[$i]." (for 循环)<br />";
```

```
    }
    foreach ($roomtypes as $room){
      echo $room."(foreach 循环)<br />";
    }
  ?>
```

运行结果如图 6-5 所示。



图 6-5　程序运行结果

【案例分析】：

（1）for 循环只进行了 0、1、2 三次；

（2）foreach 循环则列出了数组中的所有数组元素。

6.5.2　遍历一维联合索引数组

下面以遍历酒店房间类型为例对联合索引数组进行遍历。

【例 6.5】（实例文件：ch06\6.5.php）

```
  <?php
    $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'VIP 套房
'=> 368);
    foreach ($prices_per_day as $price){
      echo $price."<br />";
    }
    foreach ($prices_per_day as $key => $value){
      echo $key.":".$value." 每天。<br />";
    }
    reset($prices_per_day);
    while ($element = each($prices_per_day)){
      echo $element['key']."\t";
       echo $element['value'];
      echo "<br />";
    }
    reset($prices_per_day);
    while (list($type, $price) = each($prices_per_day)){
    echo "$type - $price<br />";
    }
```

```
?>
```

运行结果如图 6-6 所示。

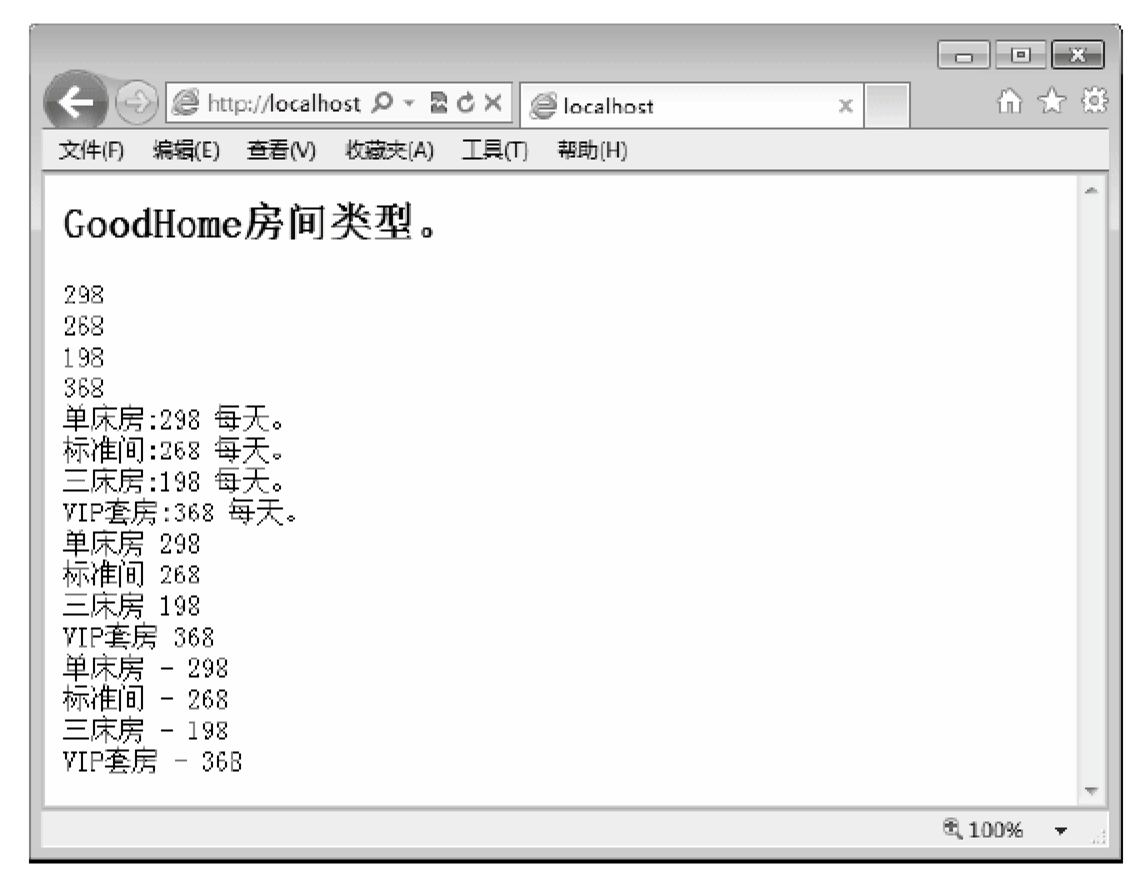


图 6-6　程序运行结果

【案例分析】：

（1）其中，foreach ($prices_per_day as $price){} 遍历数组元素，所以输出 4 个整型数字。而 foreach ($prices_per_day as $key => $value){}则除了遍历数组元素，还遍历其所对应的关键字，如 onebedroom 是数组元素 298 的关键字。

（2）这段程序中使用了 while 循环，还用到了几个新的函数，即 reset()、each()和 list()。由于在前面的代码中，$prices_per_day 已经被 foreach 循环遍历过，而内存中的实时元素为数组的最后一个元素。因此，如果想用 while 循环来遍历数组，就必须用 reset()函数把实时元素重新定义为数组的开头元素。each()则是用来遍历数组元素及其关键字的函数。list()则是把 each()中的值分开赋值和输出的函数。

6.5.3　遍历多维数组

下面以使用多维数组编写房间类型为例进行遍历，具体操作步骤如下。

【例 6.6】（实例文件：ch06\6.6.php）

```
<?php
 $roomtypes = array( array( 'type'=>'单床房',
                        'info'=>'此房间为单人单间。',
             'price_per_day'=>298
                      ),
          array( 'type'=>'标准间',
                        'info'=>'此房间为两床标准配置。',
             'price_per_day'=>268
                      ),
          array( 'type'=>'三床房',
                        'info'=>'此房间备有三张床',
             'price_per_day'=>198
```

```
                            ),
                array( 'type'=>'VIP 套房',
                            'info'=>'此房间为 VIP 两间内外套房',
                  'price_per_day'=>368
                            )
    );
    for ($row = 0; $row < 4; $row++){
      while (list($key, $value ) = each( $roomtypes[$row])){
          echo "$key:$value"."\t |";
    }
      echo '<br />';
    }
  ?>
```

运行结果如图 6-7 所示。

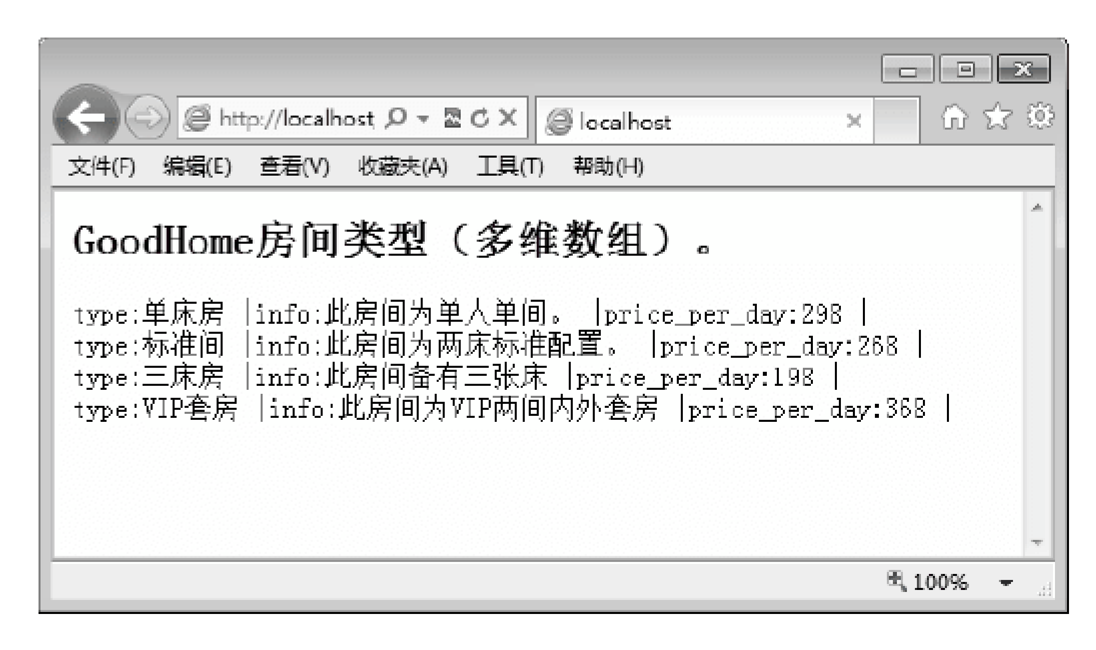


图 6-7　程序运行结果

【案例分析】：

（1）其中，$roomtypes 中的每个数组元素都是一个数组，而作为数组元素的数组又都有三个拥有键名的数组元素。

（2）使用 for 循环配合 each()、list()函数来遍历数组元素，便得到如图 6-7 所示的输出。

6.6　数组排序

本节主要讲述如何对一维和多维数组进行排序操作。

6.6.1　一维数组排序

下面通过以下实例展示如何对数组排序，具体操作步骤如下。

【例 6.7】（实例文件：ch06\6.7.php）

```
  <?php
    $roomtypes = array( '单床房','标准间','三床房','VIP 套房');
    $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'VIP 套房
'=> 368);
    sort($roomtypes);
```

```
    foreach ($roomtypes as $key => $value){
      echo $key.":".$value."<br />";
    }
    asort($prices_per_day);
    foreach ($prices_per_day as $key => $value){
      echo $key.":".$value." 每日。<br />";
    }
    ksort($prices_per_day);
    foreach ($prices_per_day as $key => $value){
      echo $key.":".$value." 每天。<br />";
    }
    rsort($roomtypes);
    foreach ($roomtypes as $key => $value){
      echo $key.":".$value."<br />";
    }
    arsort($prices_per_day);
    foreach ($prices_per_day as $key => $value){
      echo $key.":".$value." 每日。<br />";
    }
    krsort($prices_per_day);
    foreach ($prices_per_day as $key => $value){
      echo $key.":".$value." 每天。<br />";
    }
  ?>
```

运行结果如图 6-8 所示。

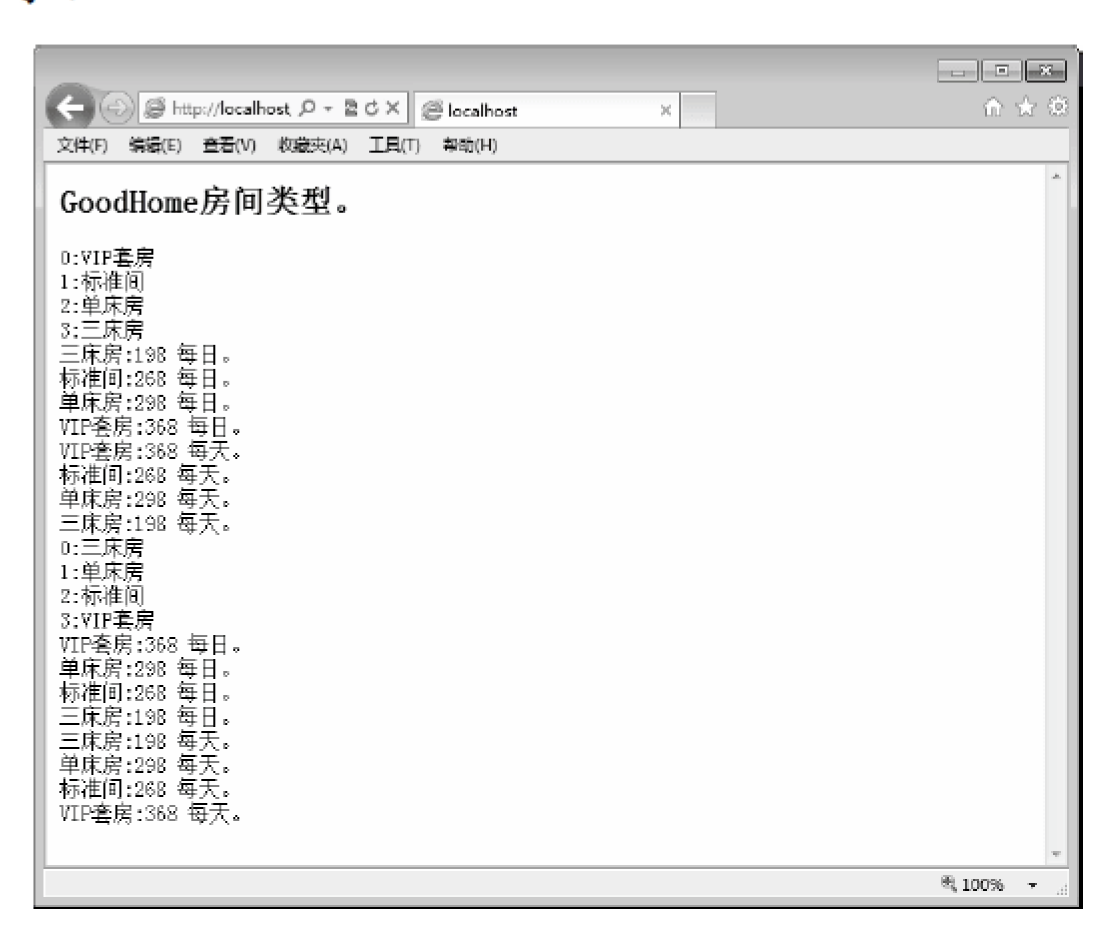


图 6-8　程序运行结果

【案例分析】：

（1）这段代码是关于数组排序的内容，涉及 sort()、asort()、ksort()、rsort()、arsort()和 krsort()。其中，sort()是默认排序。asort()根据数组元素的值的升序排序。ksort()根据数组元素的键值，也就是关键字的升序排序。

（2）rsort()、arsort()和 krsort()则正好与所对应的升序排序相反，都为降序排序。

6.6.2 多维数组排序

对于一维数组，通过 sort()等一系列的排序函数，就可以对它进行排序。而对于多维数组，排序就没有那么简单了。首先需要设定一个排序方法，也就是建立一个排序函数。再通过 usort()函数对特定数组采用特定排序方法来进行排序。下面的案例介绍多维数组排序，具体步骤如下。

【例 6.8】（实例文件：ch06\6.8.php）

```
<?php
  $roomtypes = array( array( 'type'=>'单床房',
                           'info'=>'此房间为单人单间。',
            'price_per_day'=>298
                          ),
           array( 'type'=>'标准间',
                           'info'=>'此房间为两床标准配置。',
            'price_per_day'=>268
                          ),
           array( 'type'=>'三床房',
                           'info'=>'此房间备有三张床',
            'price_per_day'=>198
                          ),
           array( 'type'=>'VIP 套房',
                           'info'=>'此房间为 VIP 两间内外套房',
            'price_per_day'=>368
                          )

  );
  function compare($x, $y){
     if ($x['price_per_day'] == $y['price_per_day']){
      return 0;
   }else if ($x['price_per_day'] < $y['price_per_day']){
      return -1;
   }else{
      return 1;
   }
  }

  usort($roomtypes, 'compare');

  for ($row = 0; $row < 4; $row++){
    reset($roomtypes[$row]);
    while (list($key, $value ) = each( $roomtypes[$row])){
        echo "$key:$value"."\t |";
  }
    echo '<br />';
  }
?>
```

运行结果如图 6-9 所示。

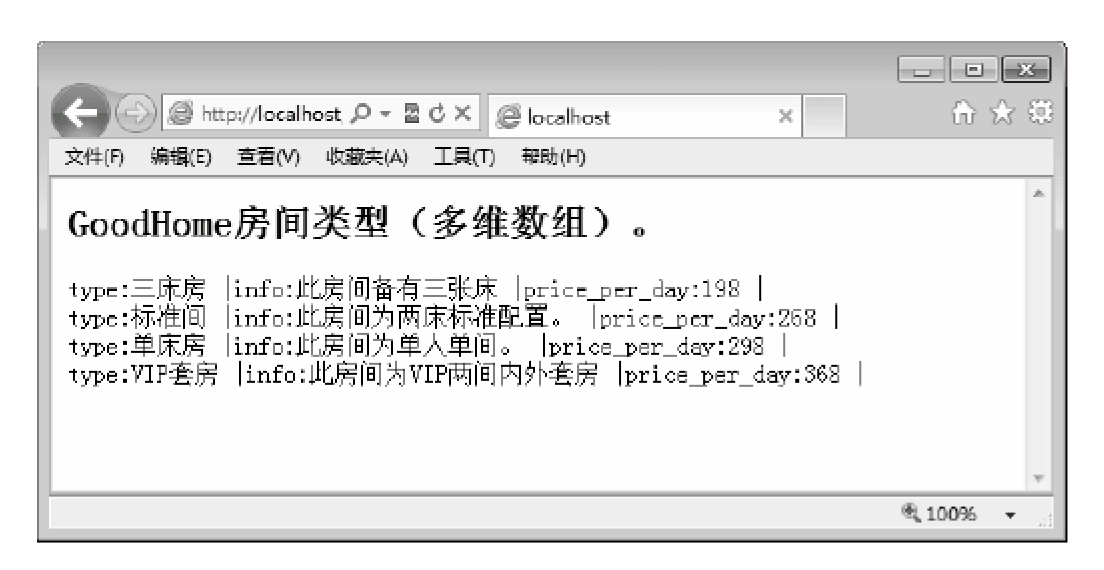


图 6-9　程序运行结果

【案例分析】：

（1）其中，函数 compare()定义了排序方法，通过对 price_per_day 这一数组元素的对比，进行排序。然后 usort()采用 compare 方法对$roomtypes 这一多维数组进行排序。

（2）如果这个排序的结果是正向排序，那么如何进行反向排序呢？这就需要对排序方法进行调整。其中，recompare()就是上一段程序中 compare()的相反判断，同样采用 usort()函数输出后，得到如图 6-9 所示的排序，正好与前一段程序输出顺序相反。

6.7　字符串与数组的转换

可使用 explode()和 implode()函数来实现字符串和数组之间的转换。Explode 用于把字符串按照一定的规则拆分为数组中的元素，并且形成数组。Implode()函数用于把数组中的元素按照一定的连接方式转换为字符串。

下面的例子介绍使用 explode()和 implode()函数来实现字符串和数组之间的转换。

【例 6.9】（实例文件：ch06\6.9.php）

```
<?php
 $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'VIP 套房
'=> 368);
 echo implode('元每天/ ',$prices_per_day).'<br />';

 $roomtypes ='单床房,标准间,三床房,VIP 套房';
  print_r(explode(',',$roomtypes));
?>
```

运行结果如图 6-10 所示。

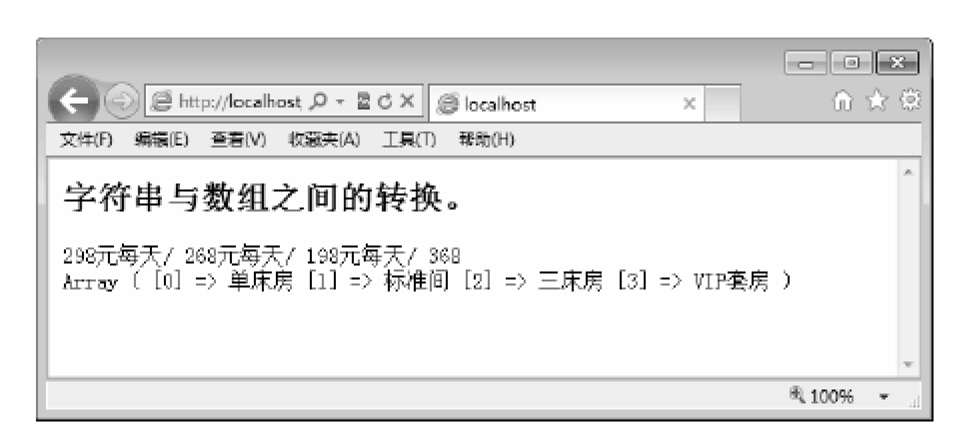


图 6-10　程序运行结果

【案例分析】：

（1）其中，$prices_per_day 为数组。implode('元每天/ ',$prices_per_day)对$prices_per_day 中的

数组元素中间添加连接内容，也叫元素胶水（glue），把它们连接成一个字符串输出。这个元素胶水（glue）只在元素之间。

（2）$roomtypes 为一个由“，”号分开的字符串。explode(',',$roomtypes)确认分隔符为“，”号后，以“，”号为标记把字符串中的字符分为 4 个数组元素，并且生成数组并返回。

6.8 向数组中添加和删除元素

数组创建完成后，用户还可以继续添加和删除元素，从而满足实际工作的需要。

6.8.1 向数组中添加元素

数组是数组元素的集合。如果向数组中添加元素，就像往一个盒子里面放东西。这就涉及“先进先出”或是“后进先出”的问题。

- 先进先出，有点像排队买火车票。先进入购买窗口区域，购买完成之后从旁边的出口出去。
- 后进先出，有点像给枪的弹夹上子弹。最后押上的那一颗子弹是要最先打出去的。

PHP 对数组添加元素的处理使用 push 、pop、shift 和 unshift 函数来实现，可以实现先进先出，也可以实现后进先出。

下面通过实例介绍在数组前面添加元素，以实现后进先出。

【例 6.10】（实例文件：ch06\6.10.php）

```
<?php
  $clients = array('李丽丽','赵大勇','方芳芳');
  array_unshift($clients, '王小明','刘小帅');
  print_r($clients);
?>
```

运行结果如图 6-11 所示。

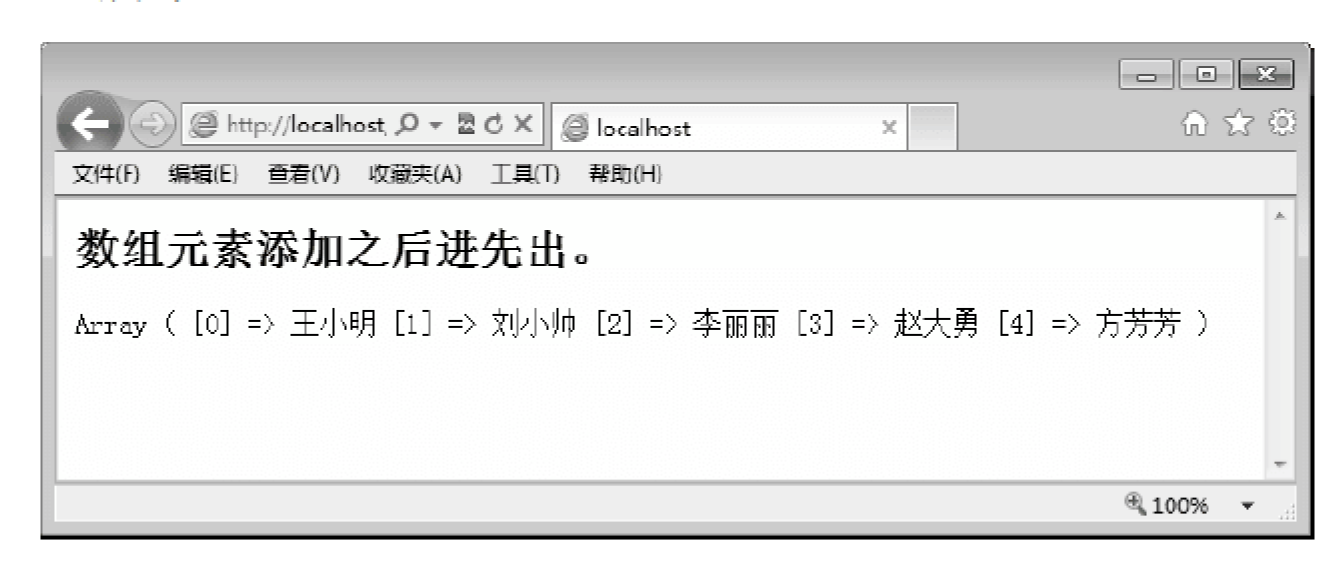


图 6-11 程序运行结果

【案例分析】：

（1）其中，数组$clients 原本拥有 3 个数组元素。 array_unshift()向数组$clients 的头部添加了数组元素'王小明'、'刘小帅'。最后通过 print_r()输出，通过其数字索引可以知道添加元素的位置。

（2）array_unshift()函数的格式为：

```
array_unshift（目标数组，[预添加数组元素，预添加数组元素，……]）
```

同样通过以下实例介绍如何在数组后面添加元素，以实现先进先出。

【例 6.11】（实例文件：ch06\6.11.php）

```
<?php
  $clients = array('李丽丽','赵大勇','方芳芳');
  array_push($clients, '王小明','刘小帅');
  print_r($clients);
?>
```

运行结果如图 6-12 所示。

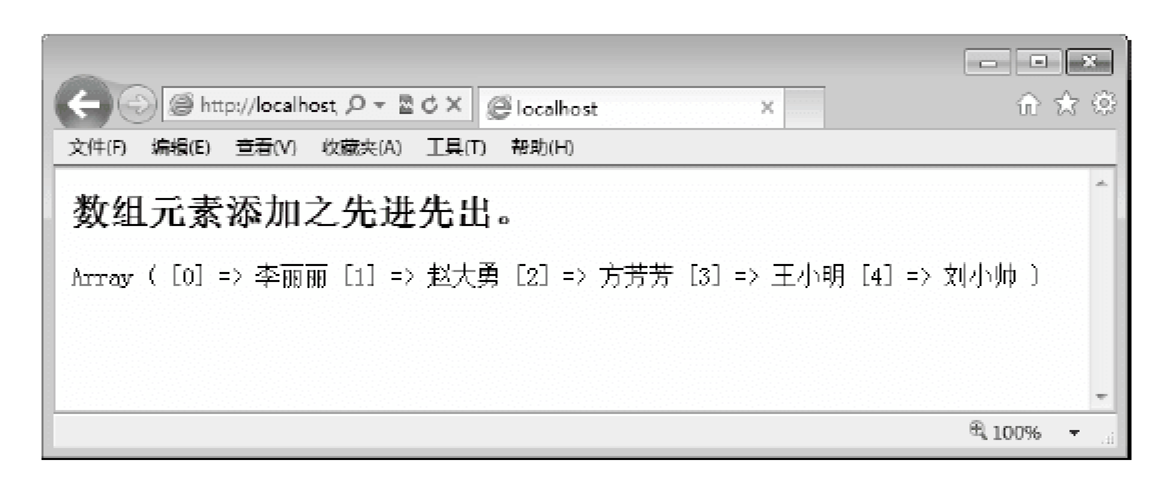


图 6-12　程序运行结果

【案例分析】：

（1）其中，数组$clients 原本拥有三个数组元素。 array_push()向数组$clients 的尾部添加了数组元素'王小明'、'刘小帅'。最后通过 print_r()输出，通过其数字索引可以知道添加元素的位置。

（2）array_push()函数的格式为：

```
array_push（目标数组，[预添加数组元素，预添加数组元素，……]）
```

push 就是“推”的意思，这个过程就像排队的时候把人从队伍后面向前推。

6.8.2　从数组中删除元素

从数组中删除元素是添加元素的逆过程。PHP 使用 array_shift()和 array_pop()函数分别从数组的头部和尾部删除元素。

下面的例子介绍如何在数组前面删除第一个元素并返回元素值。

【例 6.12】（实例文件：ch06\6.12.php）

```
<?php
  $serivces = array('洗衣','订餐','导游','翻译');
  $deletedserivces = array_shift($serivces);
  echo $deletedserivces."<br />";
  print_r($serivces);
?>
```

运行结果如图 6-13 所示。

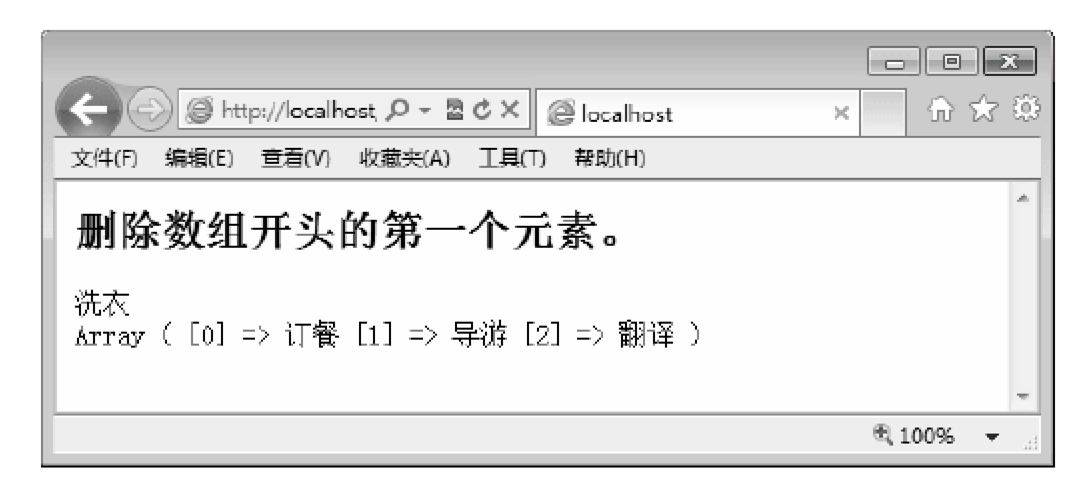


图 6-13　程序运行结果

【案例分析】：

（1）其中，数组$serivces 原本拥有 4 个数组元素。 array_shift()从数组$serivces 的头部删除了第一个数组元素，并且直接把所删除的元素值返回，且赋值给变量$deletedserivces。最后通过 echo 输出$deletedserivces，并用 print_r()输出$serivces。

（2）array_ shift()函数仅仅删除目标数组的头一个数组元素，它的格式如下：

```
array_ shift（目标数组）
```

以上为数字索引数组，如果是带键值的联合索引数组，它的效果相同，返回所删除元素的元素值。

同样，通过下面的实例介绍如何在数组后面删除最后一个元素并返回元素值。

【例 6.13】（实例文件：ch06\6.13.php）

```
<?php
  $serivces = array('s1'=>'洗衣','s2'=>'订餐','s3'=>'导游','s4'=>'翻译');
  $deletedserivces = array_pop($serivces);
  echo $deletedserivces."<br />";
  print_r($serivces);
?>
```

运行结果如图 6-14 所示。

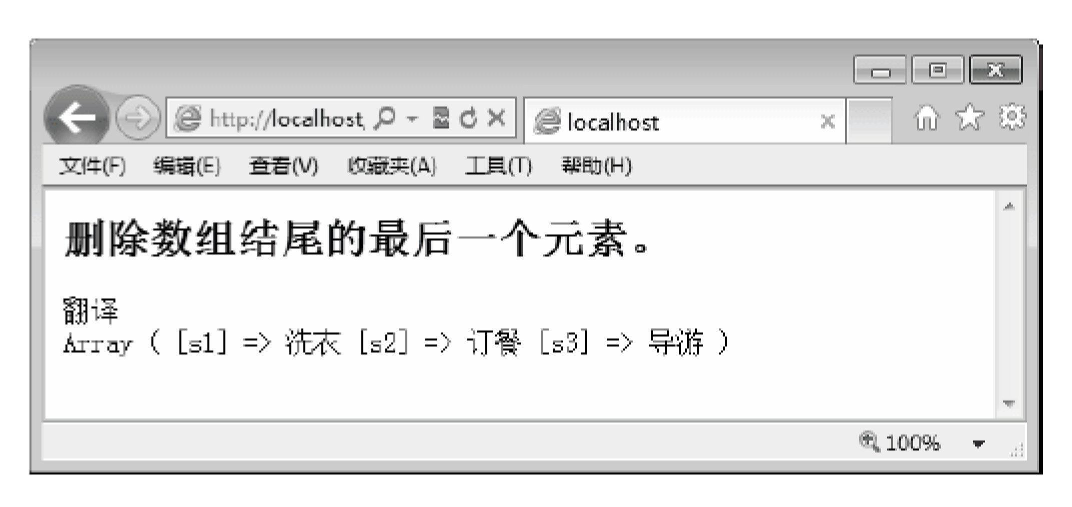


图 6-14　程序运行结果

【案例分析】：

（1）其中，数组$serivces 原本拥有 4 个数组元素。 array_pop()从数组$serivces 的尾部删除了最后一个数组元素，并且直接把所删除的元素值返回，且赋值给了变量$deletedserivces。最后通过 echo 输出$deletedserivces，以及用 print_r()输出$serivces。

（2）array_ pop()函数仅仅删除目标数组的最后一个数组元素。它的格式如下：

```
array_ pop（目标数组）
```

这个例子中的数组是一个联合数组。

6.9 查询数组中指定元素

数组是一个数据集合，能够在不同类型的数组和不同结构的数组内确定某个特定元素是否存在，是必要的。PHP 提供 in_array()、array_key_exists()、array_search()、array_keys()和 array_values()函数，按照不同方式查询数组元素。

通过下面的例子介绍如何查询数字索引数组和联合索引数组，两者都是一维数组。

【例 6.14】（实例文件：ch06\6.14.php）

```
<?php
 $roomtypes = array( '单床房','标准间','三床房','VIP 套房');
 $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'VIP 套房'=> 368);

  if(in_array( '单床房',$roomtypes)){echo '单床房元素在数组$roomtypes 中。<br />';}
  if(array_key_exists( '单床房',$prices_per_day)){echo '键名为单床房的元素在数组$prices_per_day 中。<br />';}
  if(array_search( 268,$prices_per_day)){echo '值为 268 的元素在数组$prices_per_day 中。<br />';}

$prices_per_day_keys = array_keys($prices_per_day);
print_r($prices_per_day_keys);
$prices_per_day_values = array_values($prices_per_day);
print_r($prices_per_day_values);
?>
```

运行结果如图 6-15 所示。



图 6-15 程序运行结果

【案例分析】：

（1）其中，数组$roomtypes 为一个数字索引数组。in_array('单床房',$roomtypes) 判定元素'单床房'是否在数组$roomtypes 中，如果在，则返回 true。if 语句得到返回值为真，便打印表述。

（2）数组$prices_per_day 为一个联合索引数组。array_key_exists('单床房',$prices_per_day)判定一个键值为'单床房'的元素是否在数组$prices_per_day 中，如果在，则返回 true。if 语句得到返回

值为真，便打印表述。array_key_exists( )是专门针对联合数组的“键名”进行查询的函数。

（3）array_search()是专门针对联合数组的“元素值”进行查询的函数。同样针对$prices_per_day 这个联合数组。array_search(268,$prices_per_day)判定一个元素值为 268 的元素是否在数组$prices_per_day 中，如果在，则返回 true。if 语句得到返回值为真，便打印表述。

（4）函数 array_keys()是取得数组“键值”，并把键值作为数组元素输出为一个数字索引数组的函数，主要用于联合索引数组。array_keys($prices_per_day)获得数组$prices_per_day 的键值，并把它赋值给变量$prices_per_day_keys 为一个数组。用 print_r()打印表述。函数 array_keys()虽然也可以取得数字索引数组的数字索引，但是这样做意义不大。

（5）函数 array_values()是取得数组元素的“元素值”，并把元素值作为数组元素输出为一个数字索引数组的函数。array_values($prices_per_day) 获得数组$prices_per_day 的元素值，并把它赋值给变量$prices_per_day_values，为一个数组，用 print_r()打印表述。

这几个函数只是针对一维数组，无法用于多维数组。它们在查询多维数组的时候，只会处理最外围的数组，其他内嵌的数组都作为数组元素处理，不会得到内嵌数组内的键值和元素值。

6.10 统计数组元素个数

使用 count()函数统计数组的元素个数。

下面通过实例介绍如何使用 count()函数统计数组的元素个数。

【例 6.15】（实例文件：ch06\6.15.php）

```
<?php
  $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'VIP 套房'=> 368);
  $roomtypesinfo = array( array( 'type'=>'单床房',
                         'info'=>'此房间为单人单间。',
                         'price_per_day'=>298
                       ),
                  array( 'type'=>'标准间',
                         'info'=>'此房间为两床标准配置。',
                         'price_per_day'=>268
                       ),
                  array( 'type'=>'三床房',
                         'info'=>'此房间备有三张床',
                         'price_per_day'=>198
                       ),
                  array( 'type'=>'VIP 套房',
                         'info'=>'此房间为 VIP 两间内外套房',
                         'price_per_day'=>368
                       )
                     );
  echo count($prices_per_day).'个元素在数组$prices_per_day 中。<br />';
  echo count($roomtypesinfo).'个内嵌数组在二维数组$roomtypesinfo 中。<br />';
  echo count($roomtypesinfo,1).'个元素$roomtypesinfo 中。<br />';
```

```
?>
```

运行结果如图 6-16 所示。

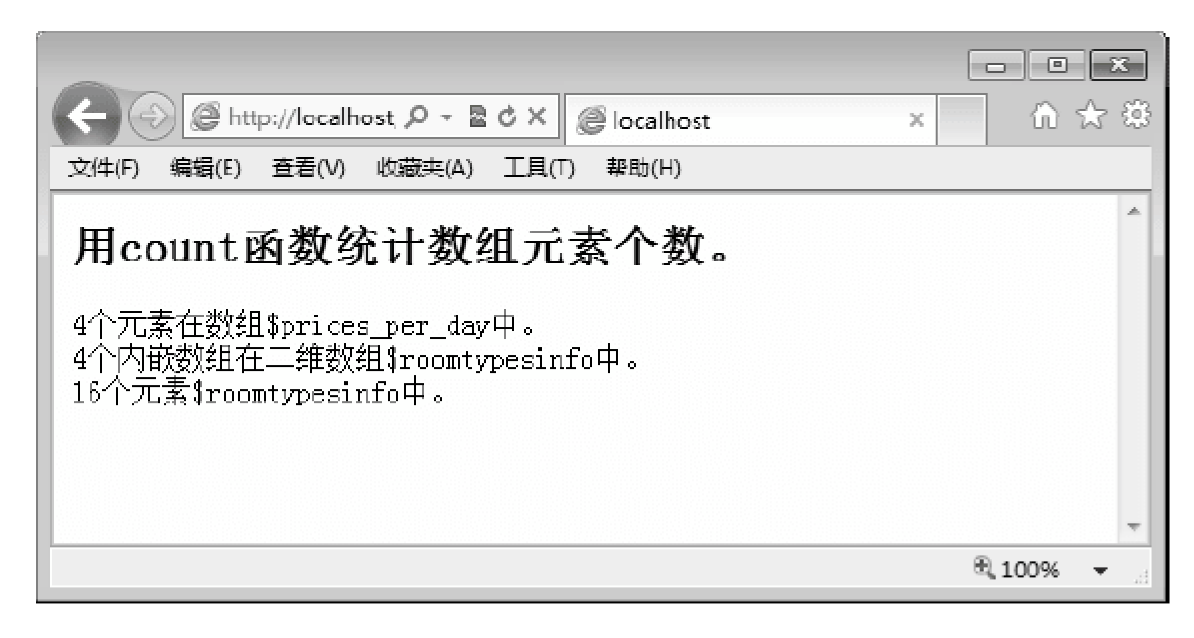


图 6-16　程序运行结果

【案例分析】：

（1）其中，数组$prices_per_day 通过 count()函数返回整数 4。因为数组$prices_per_day 有 4 个数组元素。

（2）数组$roomtypesinfo 为一个二维数组。count($roomtypesinfo)只统计了数组$roomtypesinfo 内的 4 个内嵌数组的数量，所以输出如图 6-16 所示。

（3）echo count($roomtypesinfo,1)这一语句中，count()函数设置了一个模式（mod）为整数“1”。这个模式（mod）设置为整数“1”的意义是，count 统计的时候要对数组内部所有的内嵌数组进行循环查询，所以最终的结果是所有内嵌数组的个数加上内嵌数组内元素的个数，即 4 个内嵌数组加上 12 个数组元素，为 16。

使用 array_count_values()函数对数组内的元素值进行统计，并且返回一个以函数值为“键值”、以函数值个数为“元素值”的数组。

下面通过实例介绍如何使用 array_count_values()函数统计数组的元素值个数。

【例 6.16】（实例文件：ch06\6.16.php）

```
<?php
  $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'四床房'=> 198,'VIP 套房'=> 368);
  print_r(array_count_values($prices_per_day));
?>
```

运行结果如图 6-17 所示。

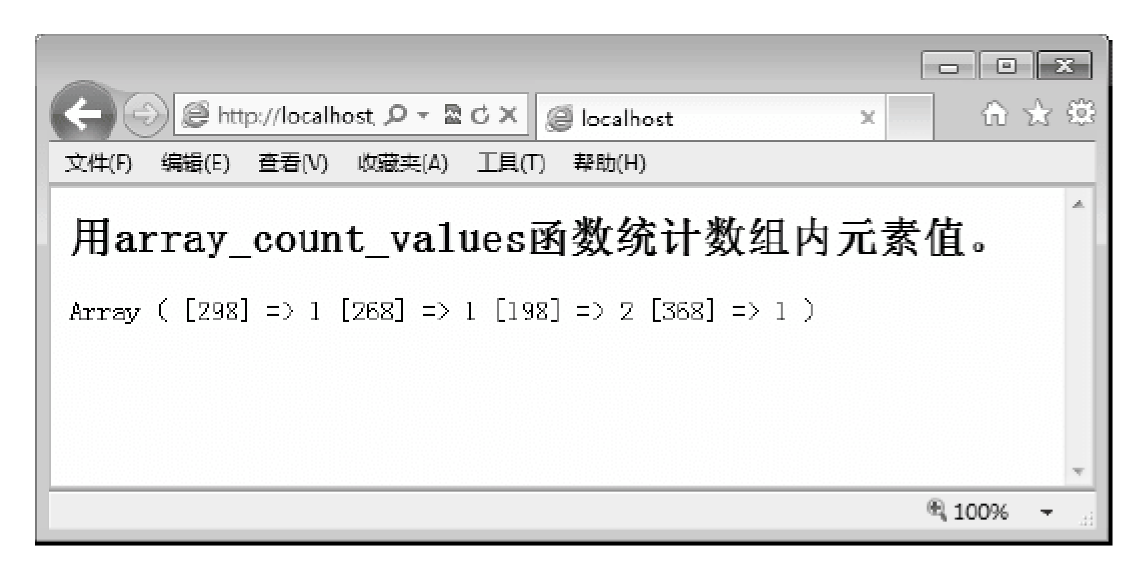


图 6-17　程序运行结果

【案例分析】：

（1）其中，数组$prices_per_day 为一个联合数组，通过 array_count_values($prices_per_day)统计数组内元素值的个数和分布，然后以键值和值的形式返回一个数组，如图 6-17 所示。元素值为 198 的元素有两个，虽然它们的键值完全不同。

（2）array_count_values()只能用于一维数组，因为它不能把内嵌的数组当作元素进行统计。

6.11 删除数组中的重复元素

可使用 array_unique()函数实现数组中元素的唯一性，也就是去掉数组中重复的元素。不管是数字索引数组还是联合索引数组，都是以元素值为准。array_unique()函数返回具有唯一性元素值的数组。

下面通过实例介绍如何使用 array_unique()函数去掉数组中重复的元素。

【例 6.17】（实例文件：ch06\6.17.php）

```
<?php
    $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'四床房'=>
198,'VIP 套房'=> 368);
    $prices_per_day2 =array('单床房'=> 298,'标准间'=> 268,'四床房'=> 198,'三床房'=>
198,'VIP 套房'=> 368);
    print_r(array_unique($prices_per_day));
    print_r(array_unique($prices_per_day2));
?>
```

运行结果如图 6-18 所示。

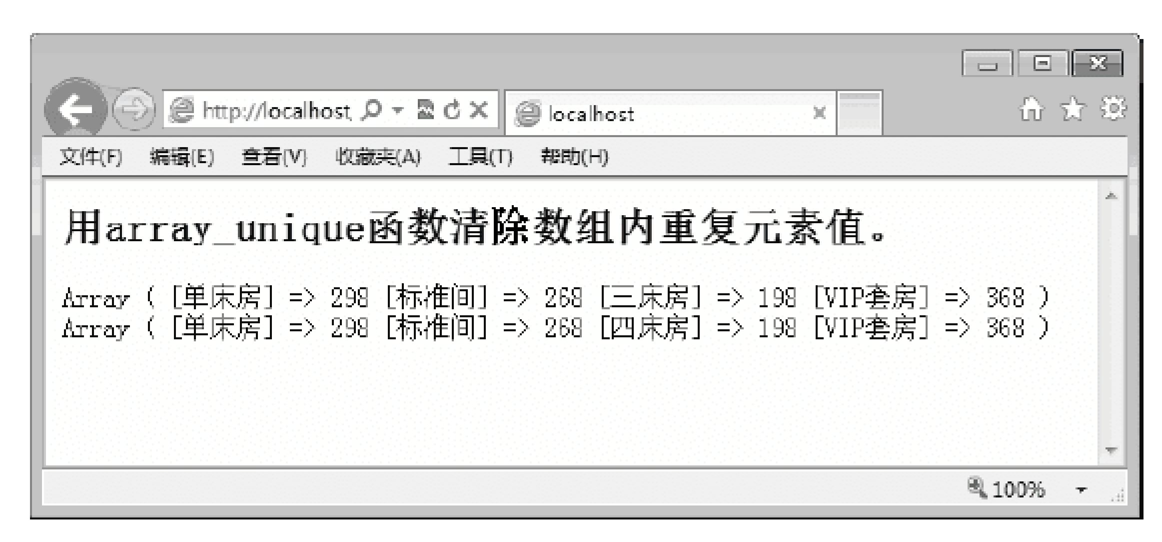


图 6-18　程序运行结果

【案例分析】：

其中，数组$prices_per_day 为一个联合索引数组，通过 array_unique ($prices_per_day)去除重复的元素值。array_unique ()函数去除重复的值就是去除第二个出现的相同值。所以，由于$prices_per_day 与$prices_per_day2 数组中，键值为“三床房”和键值为“四床方”的 198 元素的位置正好相反，所以对两次输出所保留的值也正好相反，如图 6-18 所示。

6.12 调换数组中的键值和元素值

可使用 array_flip()函数调换数组中的键值和元素值。

下面通过实例介绍如何使用 array_flip()函数调换数组中的键值和元素值，具体方法如下。

【例 6.18】（实例文件：ch06\6.18.php）

```
<?php
 $prices_per_day = array('单床房'=> 298,'标准间'=> 268,'三床房'=> 198,'四床房'=>
198,'VIP 套房'=> 368);
 print_r(array_flip ($prices_per_day));
?>
```

运行结果如图 6-19 所示。



图 6-19　程序运行结果

其中数组$prices_per_day 为一个联合索引数组，通过 array_flip ($prices_per_day)调换联合索引数组的键值和元素值，并且加以返回。但有意思的是$prices_per_day 是一个拥有重复元素值的数组，且这两个重复元素值的“键名”是不同的。array_flip ()逐个调换每个数组元素的键值和元素值。而如果原来的元素值变为键名以后，就有两个原先为键名，现在调换为元素值的数值与之对应。调换后，array_flip ()等于对原来的元素值（即现在的键名）赋值。当 array_flip ()再次调换到原来相同的、现在为键名的值时，相当于对同一个键名再次赋值，则头一个调换时的赋值将会被覆盖，显示的是第二次的赋值。

6.13　实战演练——数组的序列化

数组的序列化（serialize）用来将数组的数据转换为字符串，以方便传递和数据库的存储。而与之相对应的操作就是反序列化（unserialize），把字符串数据转换为数组加以使用。

下面通过实介绍 serialize()函数和 unserialize()函数。

【例 6.19】（实例文件：ch06\6.19.php）

```
<?php
$arr = array('王小明','李丽丽','方芳芳','刘小帅','张大勇','张明明');
$str = serialize($arr);
echo $str."<br /><br />";
$new_arr = unserialize($str);
print_r($new_arr);
?>
```

运行结果如图 6-20 所示。

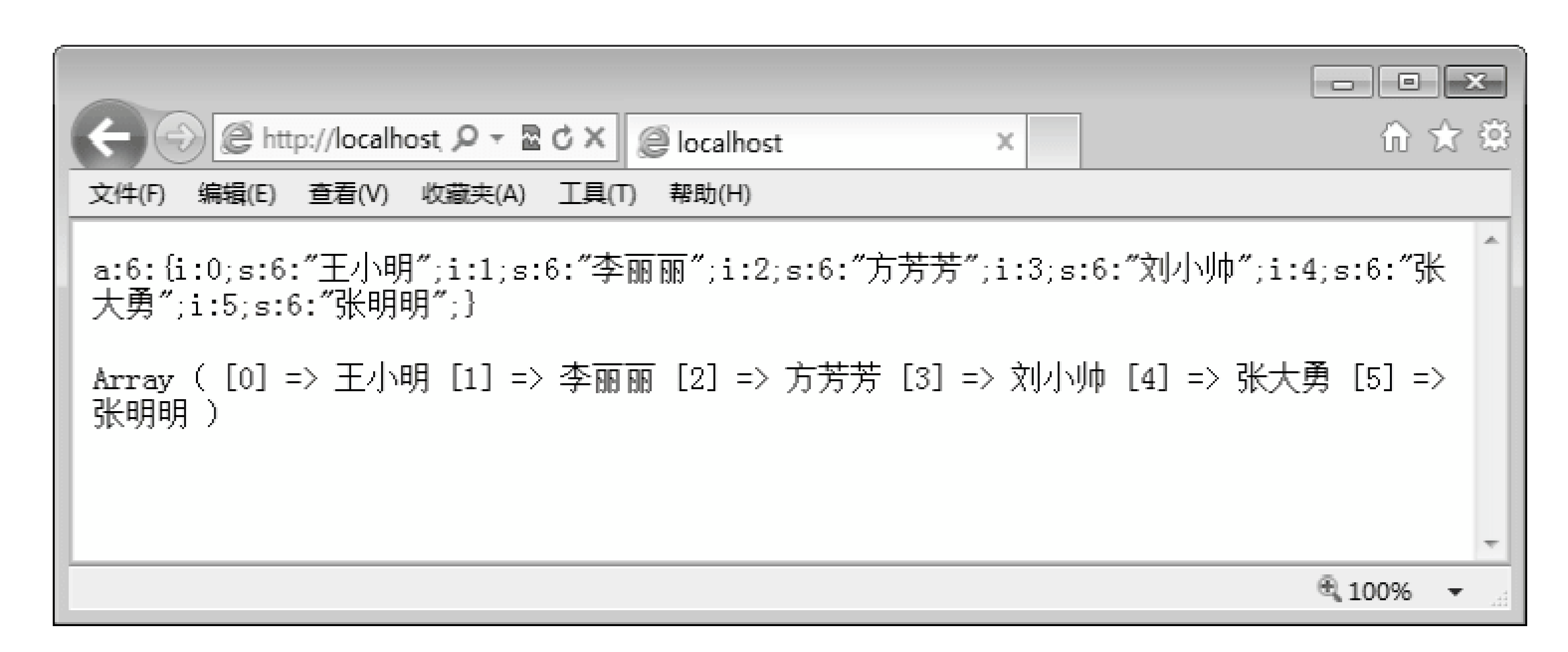


图 6-20　程序运行结果

erialize()和 unserialize()两个函数的使用比较简单，但是通过这样的方法对数组数据的存储和传递将会十分方便。比如，可以直接把序列化之后的数组数据存放在数据库的某个字段中。在使用时再通过反序列化进行处理。

6.14　高手私房菜

技巧 1：数组的合并与联合的区别是什么？

对数组的合并使用 array_merge()函数。两个数组的元素会合并为一个数组的元素。而数组的联合，是指两个一维数组，一个作为关键字，一个作为数组元素值，而联合成为一个新的联合索引数组。

技巧 2：如何快速清空数组？

在 PHP 中，快速清空数组的方法如下。

```
arr=array()      //理解为重新给变量赋一个空的数组。
unset($arr)     //这才是真正意义上的释放，将资源完全释放。
```

6.15　经典习题

（1）制作一个包含常量数组和变量数组的例子，并分析它们的区别。
（2）制作一个包含一维数组和多维数组的例子。
（3）制作一个包含遍历数组的例子。
（4）制作一个包含数组排序的例子。
（5）制作一个包含字符串和数组相互转换的例子。

第 7 章　时间和日期

时间和日期对于很多应用来说都是十分敏感的，程序中很多情况下都是依靠时间和日期才能作出判断从而完成操作的，例如在酒店商务网站中查看最新房价的情况，这和时间是密不可分的。本章将介绍日期和时间的获得及格式化方面的内容。

本章学习目标

- 了解系统时间的设置方法
- 熟悉 UNIX 时间戳的概念
- 掌握获取当前时间戳的方法
- 掌握获取当前日期和时间的方法
- 掌握使用时间戳获取日期的方法
- 掌握检验日期有效性的方法
- 掌握输出格式化日期的方法
- 掌握比较时间大小的方法
- 掌握实现倒计时功能的方法

7.1　系统时区设置

这里的系统时区是指运行 PHP 的系统环境。常见的有 Windows 系统和 UNIX-like（类 UNIX）系统，对于时区的设置关系到运行应用的时间准确性。

7.1.1　时区划分

时区的划分是一个地理概念。从本初子午线开始向东和向西各有 12 个时区。比如北京时间是东八区，美国太平洋时间是西八区。在 Windows 系统里这个操作比较简单。在时间时区的控制面板中设置就行了。在 Linux 这样的 UNIX-like 系统中需要使用命令对时区进行设置。

7.1.2　时区设置

PHP 中日期和时间的默认设置是 GMT 格林尼治时间。在使用时间日期功能之前，需要对时区进行设置。在中国，就需要使用“Asia/Hong_Kong”香港时间。

时区的设置方法主要有以下两种。

（1）修改 php.ini 文件的设置。找到“;date.timezone=”选项，将其值修改为 date.timezone = Asia/Hong_Kong，这样系统默认时间为东八区的时间。

（2）在应用程序中，直接使用函数 date_default_timezone_set()把时区设为 date_default_timezone_set("Asia/Hong_Kong")。此种方法比较灵活。

设置完成后，date()函数便可以正常使用，不会出现时差的问题。

7.2 PHP 日期和时间函数

本节开始学习 PHP 常用日期和时间函数的使用方法和技巧。

7.2.1 关于 UNIX 时间戳

在很多情况下，程序需要对日期进行比较、运算等操作。如果按照人们日常的计算方法，很容易知道 6 月 5 号和 6 月 8 号相差几天。然而，人们日常对日期的书写方式是 2012-3-8 或 2012 年 3 月 8 日星期五。这让程序如何运算呢？对于整型数据的数学运算好像对这样的描述并不容易处理。如果想知道 3 月 8 号和 4 月 23 号相差几天，则需要把月先转换为 30 天或 31 天，再对剩余天数进行加减。这是一个很麻烦的过程。

如果时间或者日期是一个连贯的整数，这样处理起来就很方便了。幸运的是，系统的时间正是以这种方式存储的，这种方式就是时间戳，也称为 UNIX 时间戳。UNIX 系统和 UNIX-like 系统把当下的时间存储为 32 位的整数，这个整数的单位是秒，而这个整数的开始时间为格林尼治时间（GMT）的 1970 年 1 月 1 日的零点整。换句话说，就是现在的时间是 GMT1970 年 1 月 1 日的零点整到现在的秒数。

由于每一秒的时间都是确定的，这个整数就像章戳一样不可改变，所以就称为 UNIX 时间戳。

时间戳在 Windows 系统下也是成立的，但是与 UNIX 系统下不同的是，Windows 系统下的时间戳只能为正整数不能为负值。所以想用时间戳表示 1970 年 1 月 1 号以前的时间是不行的。

PHP 则完全采用了 UNIX 时间戳，所以不管 PHP 在哪个系统下运行都可以使用 UNIX 时间戳。

7.2.2 获取当前时间戳

获得当前时间的 UNIX 时间戳，以用于得到当前时间。要完成此操作直接使用 time()函数即可。time()函数不需要任何参数，直接返回当前日期和时间。

【例 7.1】（实例文件：ch07\7.1.php）

```
<?php
  $t1 =time();                    //获取当前时间戳
  echo "当前时间戳为：".$t1;     //输出当前时间戳
?>
```

运行结果如图 7-1 所示。



图 7-1 程序运行结果

【案例分析】：

（1）图中的数字 1465275514 表示从 1970 年 1 月 1 日 0 点 0 分 0 秒到本程序执行时间隔的秒数。

（2）如果每隔一段时间刷新一次页面，获取的时间戳的值将会增加。这个数字会一直不断变大，即每过 1 秒，此值就会加 1。

7.2.3 获取当前日期和时间

可使用 date()函数返回当前日期，如果在 date()函数中使用参数 U 则返回当前时间的 UNIX 时间戳。如果使用参数 d 则直接返回当前月份的 01~31 号的两位数日期。

然而，date()函数有很多的参数，具体含义如表 7-1 所示。

表 7-1 date()函数的参数及其含义

参数	含义	参数	含义
a	小写 am 或 pm	A	大写 AM 或 PM
d	01~31 的日期	D	Mon 到 Sun 的简写星期
e	显示时区	E	无此参数
f	无此参数	F	月份的全拼单词
g	12 小时格式的小时数（1 到 12）	G	24 小时格式的小时数（0 到 23）
h	12 小时格式的小时数（01 到 12）	H	24 小时格式的小时数（00 到 23）
i	分钟数（01 到 60）	I	Daylight
j	一月中的天数（从 1 到 31）	J	无此参数
l	一周中天数的全拼	L	Leap year
m	月份（从 01 到 12）	M	三个字母的月份简写（从 Jan 到 Dec）
n	月份（从 1 到 12）	N	无此参数
o	无此参数	O	与格林尼治时间相差的时间
s	秒数（从 00 到 59）	S	天数的序数表达（st、nd、rd、th）
t	一个月中天数的总数（从 28 到 31）	T	时区简写
N	无此参数	U	当前的 Unix 时间戳
w	数字表示的周天（从 0-Sunday 到 6-Saturday）	W	ISO8601 标准的一年中的周数
y	无此参数	Y	四位数的公元纪年（从 1901 到 2038）
z	一年中的天数（从 0 到 364）	Z	以秒表现的时区（从-43200 到 50400）

7.2.4 使用时间戳获取日期信息

如果相应的时间戳已经存储在数据库中，程序需要把时间戳转化为可读的日期和时间，才能满足应用的需要。PHP 中提供了 data()和 getdate()等函数来实现从时间戳到通用时间的转换。

1. data()函数

data()函数主要是将一个 UNIX 时间戳转化为指定的时间/日期格式。该函数的格式如下。

```
srting data(string format [时间戳整数])
```

此函数将会返回一个字符串。该字符串就是一个指定格式的日期时间，其中 format 是一个字符串，用来指定输出的时间格式。时间戳整数可以为空，如果为空，则表示为当前时间的 UNIX 时间戳。

format 参数由指定的字符构成，具体字符的含义如表 7-2 所示。

表 7-2　format 参数及含义

format 字符	含义
a	am 或 pm
A	AM 或 PM
d	几日，两位数字，若不足两位则前面补零，例如 01 至 31
D	星期几，三个英文字母，例如: Fri
F	月份，英文全名，例如: January
h	12 小时制的小时，例如: 01 至 12
H	24 小时制的小时，例如: 00 至 23
g	12 小时制的小时，不足二位不补零，例如: 1 至 12
G	24 小时制的小时，不足二位不补零，例如: 0 至 23
i	分钟。例如: 00 至 59
j	几日，二位数字，若不足二位不补零，例如: 1 至 31
l	星期几，英文全名。例如: Friday
m	月份，二位数字，若不足二位则在前面补零，例如: 01 至 12
n	月份，二位数字，若不足二位则不补零，例如: 1 至 12
M	月份，三个英文字母，例如: Jan
s	秒。例如: 00 至 59
S	字尾加英文序数，二个英文字母，例如: th，nd
t	指定月份的天数，例如: 28 至 31
U	总秒数
w	数字型的星期几，例如: 0(星期日)至 6(星期六)
Y	年，四位数字，例如: 1999
y	年，二位数字，例如: 99
z	一年中的第几天，例如: 0 至 365

下面通过实例来理解 format 字符的使用方法。

【例 7.2】（实例文件：ch07\7.2.php）

```
<?php date_default_timezone_set("PRC");
  //定义一个当前时间的变量
  $tt =time();
  echo "目前的时间为：<br />";
  //使用不同的格式化字符测试输出效果
  echo date ("Y 年 m 月 d 日[l]H 点 i 分 s 秒",$tt)."<br />";
  echo date ("y-m-d h:i:s a",$tt)."<br />";
  echo date ("Y-M-D H:I:S A",$tt)."<br />";
  echo date ("F,d,y l",$tt)." <br />";
  echo date ("Y-M-D H:I:S",$tt)." <br />";
?>
```

运行结果如图 7-2 所示。

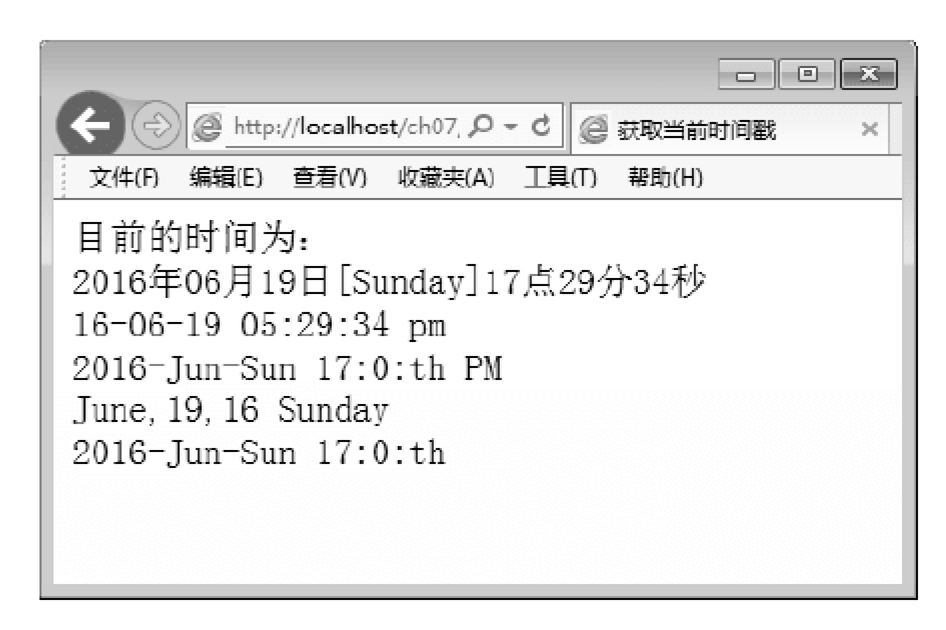


图 7-2 程序运行结果

【案例分析】：

（1）date_default_timezone_set("PRC")语句的作用是设置默认时区为北京时间。如果不设置将会显示安全警告信息。

（2）格式化字符的使用方法非常灵活，只要设置字符串中包含的字符，date()函数就能将字符串替换成指定的时间日期信息。利用上面的函数可以随意输出自己需要的日期。

2. getdate()函数

getdate()函数用于获取详细的时间信息，函数的格式如下：

```
array getdate（时间戳整数）
```

getdate()函数返回一个数组，包含日期和时间的各个部分。如果它的参数时间戳整数为空，则表示直接获取当前时间戳。

下面通过实例说明此函数的使用方法和技巧。

【例 7.3】（实例文件：ch07\7.3.php）

```
<?php date_default_timezone_set("PRC");
  //定义一个时间的变量
  $tm ="2012-08-08 08:08:08";
  echo "时间为：". $tm."<br />";
  //将格式转化为 Unix 时间戳
  $tp =strtotime($tm);
  echo "此时间的 Unix 时间戳为：".$tp."<br />";
  $ar1 =getdate($tp);
  echo "年为：". $ar1["year"]."<br />";
  echo "月为：". $ar1["mon"]."<br />";
  echo "日为：". $ar1["mday"]."<br />";
  echo "点为：". $ar1["hours"]."<br />";
  echo "分为：". $ar1["minutes"]."<br />";
  echo "秒为：". $ar1["seconds"]."<br />";
?>
```

运行结果如图 7-3 所示。

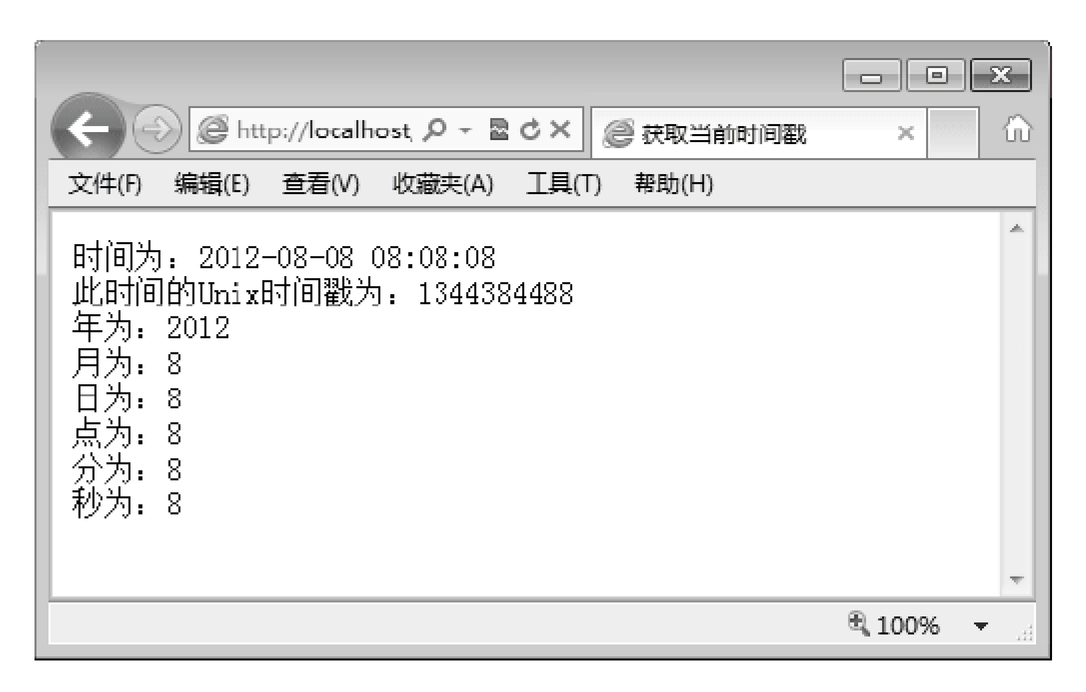


图 7-3　程序运行结果

7.2.5　检验日期的有效性

使用用户输入的时间数据时，会由于用户输入的数据不规范而导致程序运行出错。为了检查时间的合法有效性，需要使用 checkdate()函数对输入日期进行检测。它的格式为：

```
checkdate（月份，日期，年份）
```

此函数检查的项目是，年份是否在 0~32767 之间，月份是否在 1~12 之间，日期是否在相应的月份的天数内。下面通过实例来讲述如何检查日期的有效性。

【例 7.4】（实例文件：ch07\7.4.php）

```
<?php
if(checkdate(2,31,2016)){
   echo "这不可能。";
}else{
   echo "2月没有31号。";
}
?>
```

运行结果如图 7-4 所示。



图 7-4　程序运行结果

7.2.6　输出格式化时间戳的日期和时间

可使用 strftime()把时间戳格式化为日期和时间。它的格式如下：

```
strftime（格式，时间戳）
```

其中“格式”决定了如何把其后面的时间戳格式化并且输出。如果时间戳为空，则系统当前时间戳将会被使用。

关于“格式”的代码，如表 7-3 所示。

表 7-3 “格式”的代码描述

代码	描述	代码	描述
%a	周日期（缩简）	%A	周日期
%b 或%h	月份（缩简）	%B	月份
%c	标准格式的日期和时间	%C	世纪
%d	月日期（从 01 到 31）	%D	日期的缩简格式（mm/dd/yy）
%e	包含两个字符的字符串月日期（从‘01’到‘31’）	%G	根据周数的年份（4 个数字）
%g	根据周数的年份（2 个数字）	%H	小时数（从 00 到 23）
%j	一年中的天数（从 001 到 366）	%I	小时数（从 1 到 12）
%m	月份（从 01 到 12）	%M	分钟（从 00 到 59）
%n	新一行（同\n）	%P	am 或 pm
%p	am 或 pm	%R	时间使用 24 小时制表示
%r	时间使用 am 或 pm 表示	%S	秒（从 00 到 59）
%t	Tab（同\t）	%T	时间使用 hh:ss:mm 格式表示
%u	周天数（从 1-Monday 到 7-Sunday）	%U	一年中的周数（从第一周的第一个星期天开始）
%w	周天数（从 0-Sunday 到 6-Saturday）	%V	一年中的周数（以至少剩余四天的这一周开始为第一周）
%x	标准格式日期（无时间）	%W	一年中的周数（从第一周的第一个星期一开始）
%y	年份（2 个字符）	%X	标准格式时间（无日期）
%z 和%Z	时区	%Y	年份（4 个字符）

下面通过实例介绍该函数的用法。

【例 7.5】（实例文件：ch07\7.5.php）

```
<?php
  date_default_timezone_set("PRC");
  echo(strftime("%b %d %Y %X", mktime(20,0,0,12,31,98)));
  echo(gmstrftime("%b %d %Y %X", mktime(20,0,0,12,31,98)));
  //输出当前日期、时间和时区
  echo(gmstrftime("It is %a on %b %d, %Y, %X time zone: %Z",time()));
?>
```

运行结果如图 7-5 所示。

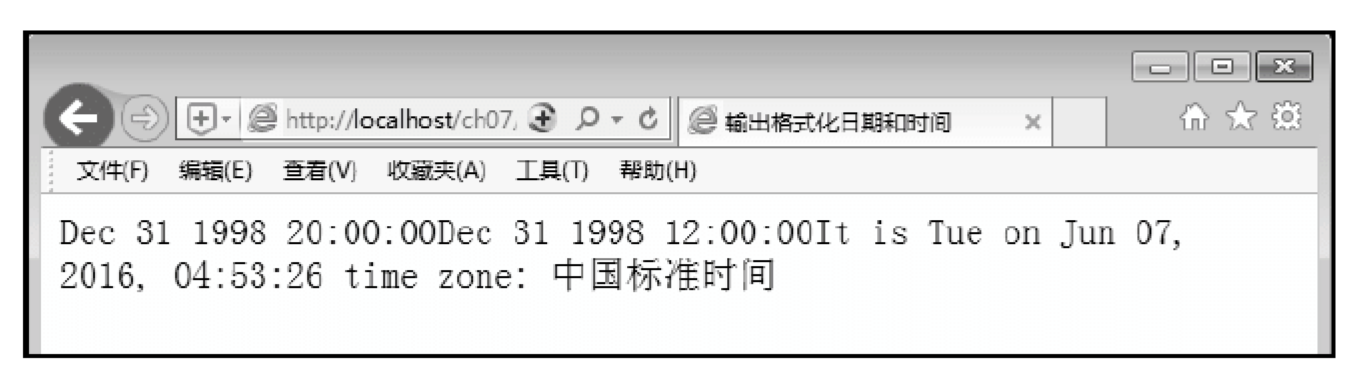


图 7-5 程序运行结果

7.2.7 显示本地化的日期和时间

由于世界上有不同的显示习惯和规范，所以日期和时间也会根据不同的地区显示为不同的形式。这就是日期和时间的本地化显示。

实现此操作需要使用到 setlocale()和 strftime()两个函数。后者已经介绍过。

可使用 setlocale()函数来改变 PHP 的本地化默认值，实现本地化的设置。它的格式为：

```
setlocale（目录，本地化值）
```

（1）“本地化值”是一个字符串，它有一个标准格式：language_COUNTRY.chareacterset。比如，想把本地化设为美国，按照此格式为 en_US.utf8；如果想把本地化设为英国，按照此格式为 en_GB.utf8；如果想把本地化设为中国，且为简体中文，按照此格式为 zh_CN.gb2312，或者 zh_CN.utf8。

（2）“目录”是指 6 个不同的本地化目录，如表 7-4 所示。

表 7-4　本地化目录

目录	含义
LC_ALL	为后面其他的目录设定本地化规则的目录
LC_COLLATE	字符串对比目录
LC_CTYPE	字母划类和规则
LC_MONETARY	货币表示规则
LC_NUMERIC	数字表示规则
LC_TIME	日期和时间表示规则

由于这里要对日期时间进行本地话设置，需要使用到的目录是 LC_TIME。下面通过实例对日期时间本地化进行讲解。

【例 7.6】（实例文件：ch07\7.6.php）

```
<?php
    date_default_timezone_set("PRC");
    date_default_timezone_set("Asia/Hong_Kong");  //设置时区为中国时区
    setlocale(LC_TIME, "zh_CN.gb2312");         //设置时间的本地化显示方式
    echo strftime("%z");                       //输出所在的时区
?>
```

运行结果如图 7-6 所示。

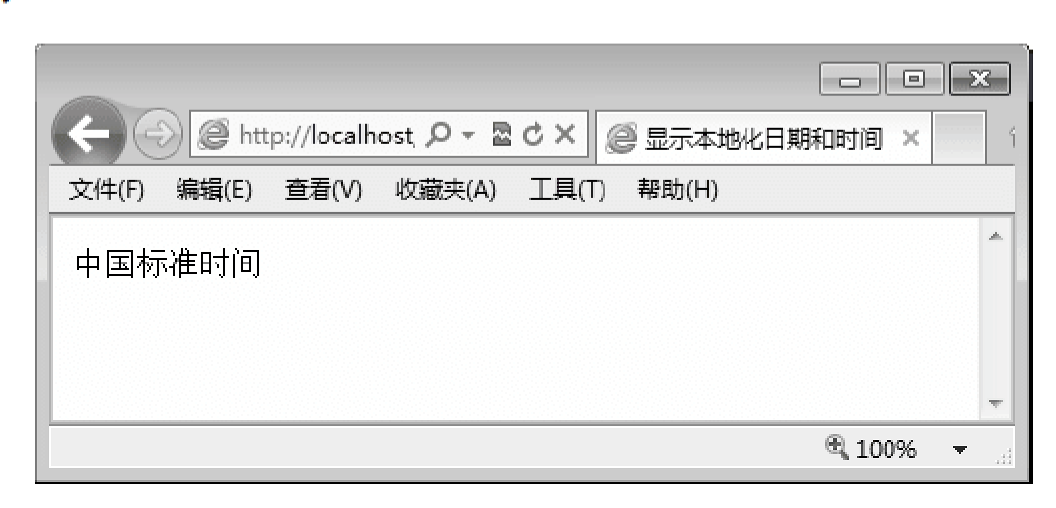


图 7-6　程序运行结果

【案例分析】：

（1）其中，date_default_timezone_set("Asia/Hong_Kong")设定时区为中国时区。
（2）setlocale(LC_TIME, "zh_CN.gb2312")设置时间的本地化显示方式为简体中文方式。
（3）strftime("%z")返回所在时区，其在页面中显示为简体中文。

7.2.8　将日期和时间解析为 UNIX 时间戳

使用给定的日期和时间，操作 mktime()函数可以生成相应的 UNIX 时间戳。它的格式为：

```
mktime（小时，分钟，秒，月份，日期，年份）
```

把相应的时间和日期的部分输入相应位置的参数，即可得到相应的时间戳。下面通过实例介绍此函数的应用方法和技巧。

【例 7.7】（实例文件：ch07\7.7.php）

```
  <?php
$timestamp = mktime(0,0,0,3,31,2012);  //获取指定日期的时间戳
echo $timestamp;                        //输出时间戳
  ?>
```

运行结果如图 7-7 所示。

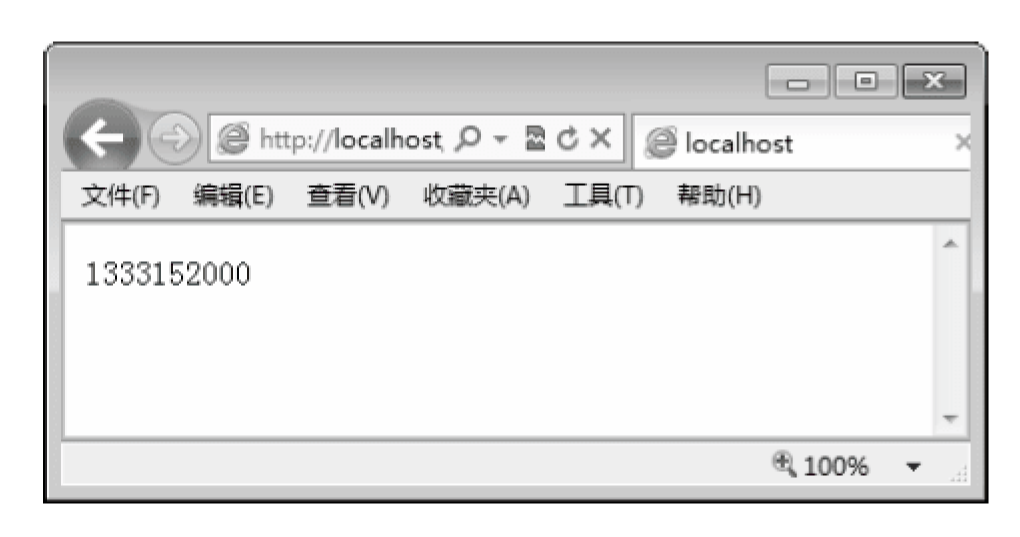


图 7-7　程序运行结果

其中 mktime(0,0,0,3,31,2012)使用的时间是 2012 年 3 月 31 号 0 点整。

7.2.9　日期和时间在 PHP 和 MySQL 数据格式之间的转换

日期和时间在 MySQL 中是按照 ISO8601 格式存储的。这种格式要求以年份打头，如 2012-03-08。从 MySQL 读取的默认格式也是这种格式。这种格式中国人是比较熟悉的。在中文应用中，几乎可以不用转换，就直接使用这种格式。

但是，在西方的表达方法中经常把年份放在月份和日期的后面，如 March 08, 2012。所以，在接触到国际，特别是符合英语使用习惯的项目时，需要把 ISO8601 格式的日期时间加以合适的转换。

有意思的是，为了解决这个英文使用习惯和 ISO8601 格式冲突的问题，MySQL 提供了把英文使用习惯的日期时间转换为符合 ISO8601 标准的两个函数，即 DATE_FOMAT()和 UNIX_TIMESTAMP()。这两个函数在 SQL 语言中使用，它们的具体用法将在介绍 MySQL 的部分详述。

7.3　实战演练 1——比较两个时间的大小

比较两个时间的大小时，如果通过一定形式的时间和日期进行比较，或者不同格式的时间和日期进行比较，都不方便。最方便的方法是把所有格式的时间都转换为时间戳，然后比较时间戳的大小。

下面通过实例来比较两个时间的大小。

【例 7.8】（实例文件：ch07\7.8.php）

```
  <?php
$timestampA = mktime(0,0,0,3,31,2015);    //获取指定日期的时间戳
$timestampB = mktime(0,0,0,1,31,2016);
if($timestampA > $timestampB ){          //比较两个时间戳的大小
  echo "2015年三月的时间戳数值大于2016年一月的。";
}elseif( $timestampA < $timestampB ){
  echo "2015年三月的时间戳数值小于2016年一月的。";
}else{
     echo "两个时间相同。";
  }
  ?>
```

运行结果如图 7-8 所示。



图 7-8　程序运行结果

7.4　实战演练 2——实现倒计时功能

对于未来的时间点实现倒计时，其实就是使用现在的当下时间戳和未来的时间点进行比较和运算。

下面通过实例来介绍如何实现倒计时功能。

【例 7.9】（实例文件：ch07\7.9.php）

```
  <?php
$timestampfuture = mktime(0,0,0,12,12,2016);     //获取指定日期的时间戳
$timestampnow = time();                       //获取当前日期的时间戳
$timecount = $timestampfuture - $timestampnow;   //获取倒计时的时间差
$days = round($timecount/86400);               //获取天数
echo "今天是".date('Y F d')." ,距离2016年12月12号的时间戳，还有".$days."天。";
  ?>
```

运行结果如图 7-9 所示。



图 7-9 程序运行结果

【案例分析】：

（1）其中，mktime()不带任何参数，所生成的时间戳是当前时间的时间戳。

（2）$timecount 是现在的时间戳距离未来时间点的时间戳的秒数。

（3）round($timecount/86400)，其中 86400 为一天的秒数，$timecount/86400 得到天数，round()函数取约数，得到天数。

7.5 高手私房菜

技巧 1：如何使用微秒？

有些时候，某些应用要求使用比秒更小的时间单位来表明时间。比如在一段测试程序运行的程序中，可能要使用到微秒级的时间单位来表明时间。这时只需要使用函数 microtime(true)，如下所示：

```
<?php
$timestamp = microtime(true);
echo $timestamp;
?>
```

返回的结果为 1315560215.7656。时间戳精确到小数点后 4 位。

技巧 2：定义时间和日期时出现警告怎么办？

在运行 PHP 程序时，有时会出现这样的警告：PHP Warning: date(): It is not safe to rely on the system's timezone settings。出现上述警告是因为 PHP 所取的时间是格林尼治标准时间，和用户当地的时间会有出入，由于格林威治标准时间和北京时间大概差 8 个小时，所以会弹出警告。可以使用下面方法中的任意一个来解决。

（1）在页头使用 date_default_timezone_set()设置默认时区为北京时间，即 <?php date_default_timezone_set("PRC"); ?>即可，例如本章例 7.2 中所示。

（2）在 php.ini 中设置 date.timezone 的值为 PRC，设置语句为 date.timezone=PRC，同时取消这一行代码的注释，即去掉前面的分号。

7.6 经典习题

（1）制作一个包含获取当前时间戳的例子。

（2）制作一个输出当前时间和日期的例子。

（3）制作一个包含比较两个时间大小的例子。

（4）制作一个实现倒计时功能的例子。

第 8 章　面向对象编程

面向对象是现在编程的主流技术，PHP 也不例外。面向对象（object-oriented）不同于面向过程（process-oriented），它用类、对象、关系、属性等一系列概念来提高编程的效率。它主要的特性是可封装性、可继承性和多态性。本章主要讲述面向对象的相关知识。

本章学习目标

- 了解类和对象的基本概念
- 熟悉 PHP 中类的基本操作
- 掌握构造方法和析构方法
- 掌握访问方法
- 掌握类的继承
- 掌握抽象类和接口的使用方法
- 掌握面向对象的多态性

8.1　类和对象的介绍

面向对象编程的主要优势就是把编程的重心从处理过程转移到了对现实世界实体的表达。这十分符合人们的思维方式。

类（classes）和对象（objects）并不难理解。试想一下，在日常生活中，自然人对事物的认识，一般是由看到的、感受到的实体（日常生活中的吃穿住用）归纳出来，或者抽象出它们的类。比如当看到楼下停的汽车中都是 Polo 或 QQ 的时候，人们自然会想到，这些都是“两厢车”。当衣柜里到处都是 NIKE、addidas 的时候，人们同样会想到，这些都是“运动装”。“两厢车”、“运动装”就是抽象出的类。这就是人们认识世界的过程。

然而程序员需要在计算机的世界中再造一个虚拟的“真实世界”。那么，在这里程序员就要像“造物主”一样思考，就是要先定义“类”，然后再由“类”产生一个个“实体”，也就是一个个“对象”。

请考虑这样的情况。过年的时候，有的地方要制作“点心”，点心一般会有鱼、兔、狗等生动的形状。而这些不同的形状是由不同的“模具”做出来的。那么，在这里鱼、兔、狗的一个个不同的点心就是实体，最先刻好的“模具”就是类。要明白，这个“模具”指的是被刻好的“形状”，而不是制作“模具”的材料。如果你能像造物主一样用意念制作出一个个点心，那么，你的意念的“形状”就是“模具”。

OOP 是面向对象编程（Object-Oriented Programming）的缩写。对象（object）在 OOP 中是由属性和操作组成的。属性（attributes）就是对象的特性或与对象关联的变量。操作（operation）就是对象中的方法（method）或函数（function）。

由于 OOP 中最为重要的特性之一就是可封装性，所以对对象内部数据的访问，只能通过对象

的“操作”来完成，这也被称为对象的“接口”（interfaces）。因为类是对象的模板，所以类描述了它的对象的属性和方法。

另外，面向对象编程具有 3 大特点。

（1）封装性。将类的使用和实现分开管理，只保留类的接口，这样开发人员就不用知道类的实现过程，只需要知道如何使用类即可，从而大大地提高了开发的效率。

（2）继承性。“继承”是面向对象软件技术中的一个概念。如果一个类 A 继承自另一个类 B，就把这个 A 称为“B 的子类”，而把 B 称为“A 的父类”。继承可以使得子类具有父类的各种属性和方法，而不需要再次编写相同的代码。在令子类继承父类的同时，可以重新定义某些属性，并重写某些方法，即覆盖父类的原有属性和方法，使其获得与父类不同的功能。另外，还可以为子类追加新的属性和方法。继承可以实现代码的可重用性，简化对象和类的创建过程。另外，PHP 支持单继承，也就是一个子类只能有一个父类。

（3）多态性。多态是面向对象程序设计的重要特征之一，是扩展性在“继承”之后的又一重大表现。同一操作作用于不同的类的实例，将产生不同的执行结果，即不同类的对象收到相同的消息时，得到不同的结果。

8.2 PHP 中类的操作

类是面向对象中最为重要的概念之一，是面向对象设计中最基本的组成模块。可以将类简单地看作一种数据结构，在类中的数据和函数称为类的成员。

8.2.1 类的声明

在 PHP 中，声明类的关键字是 class，声明格式如下：

```
<?php
    权限修饰符  class 类名{
        类的内容;
      }
?>
```

其中权限修饰符是可选项，常见的修饰符包括 public、private 和 protected。创建类时，可以省略权限修饰符，此时默认的修饰符为 public。public、private 和 protected 的区别如下。

（1）一般情况下，属性和方法的默认项是 public，这意味着属性和方法的各个项，从类的内部和外部都可以访问。

（2）用关键字 private 声明的属性和方法，则只能从类的内部访问，也就是，只有类内部的方法才可以访问用此关键字声明的类的属性和方法。

（3）用关键字 protected 声明的属性和方法，也是只能从类的内部访问，但是，通过“继承”而产生的“子类”，也是可以访问这些属性和方法的。

例如，定义一个学生为公共类，代码如下。

```
public class Student
```

```
{
    //类的内容
}
```

8.2.2 成员属性

成员属性是指在类中声明的变量。在类中可以声明多个变量，所以对象中可以存在多个成员属性，每个变量将存储不同的对象属性信息。

例如，以下定义：

```
public class Student
{
    Public $name;//类的成员属性
}
```

其中成员属性必须使用关键词进行修饰，常见的关键词包括 public、protected 和 private。如果没有特定的意义，仍然需要用 var 关键词修饰。另外，在声明成员属性时可以不进行赋值操作。

8.2.3 成员方法

成员方法是指在类中声明的函数。在类中可以声明多个函数，所以对象中可以存在多个成员方法。类的成员方法可以通过关键字进行修饰，从而控制成员方法的使用权限。

以下是定义成员方法的例子：

```
class Student
{
    Public $name;            //类的成员属性
    function GetIp(){
                        //方法的内容;
    }
}
```

8.2.4 类的实例化

面向对象编程的思想是一切皆为对象。类是对一个事物抽象出来的结果，因此，类是抽象的。对象是某类事物中具体的那个，因此，对象就是具体的。例如，学生就是一个抽象概念，即学生类，但是姓名叫张三的就是学生类中一个具体的学生，即对象。

类和对象可以描述为如下关系。类用来描述具有相同数据结构和特征的“一组对象”，“类”是“对象”的抽象，而“对象”是“类”的具体实例，即一个类中的对象具有相同的“型”，但其中每个对象却具有各不相同的“值”。



类是具有相同或相仿结构、操作和约束规则的对象组成的集合，而对象是某一类的具体化实例，每一个类都是具有某些共同性的对象的抽象。

类的实例化格式如下：

```
$变量名=new 类名称([参数]);      //类的实例化
```

其中，new 为创建对象的关键字，$变量名返回对象的名称，用于引用类中的方法。参数是可选的，如果存在参数，则它用于指定类的构造方法或用于初始化对象的值，如果没有定义构造函数参数，PHP 会自动创建一个不带参数的默认构造函数。

如下面的例子所示：

```
class Student
{
    Public $name;            //类的成员属性
    function GetIp(){
       //方法的内容;
    }
}
$lili=new 类名称();             //类的实例化
$liufei=new 类名称();           //类的实例化
$zhangming=new 类名称();     //类的实例化
$wangyi=new 类名称();        //类的实例化
```

上面的例子实例化了 4 个对象，并且这 4 个对象之间没有任何联系，只能说明它们源于同一个类。可见，一个类可以实例化多个对象，每个对象都是独立存在的。

8.2.5 访问类中的成员属性和方法

通过对象的引用可以访问类中的成员属性和方法，这里需要使用特殊的运算符：“->”。具体的语法格式如下：

```
$变量名=new 类名称();          //类的实例化
$变量名->成员属性=值;          //为成员属性赋值
$变量名->成员属性;             //直接获取成员的属性值
$变量名->成员方法;             //访问对象中指定的方法
```

另外，用户还可以使用一些特殊的访问方法。

1. $this

$this 存在于类的每一个成员方法中，它是一个特殊的对象引用方法。成员方法属于哪个对象，$this 引用就代表哪个对象，主要作用是专门完成对象内部成员之间的访问。

2. 操作符“::”

操作符“::”可以在没有任何声明实例的情况下访问类中的成员。使用的语法格式如下：

```
关键字::变量名/常量名/方法名
```

其中关键字主要包括 parent、self 和类名 3 种。parent 关键字表示可以调用父类中的成员变量、常量和成员方法。self 关键字表示可以调用当前类中的常量和静态成员。类名关键字表示可以调用本类中的常量变量和方法。

下面通过实例介绍类的声明和实例的生成。

【例 8.1】（实例文件：ch08\8.1.php）

```
<?php
  //定义类
class guests{
    private $name;
    private $gender;
    function setname($name){
        $this->name = $name;
    }
  //定义函数
    function getname(){
         return $this->name;
    }
    function setgender($gender){
        $this->gender = $gender;
    }
    function getgender(){
         return $this->gender;
    }
  };
  $xiaoming = new guests;   //生成实例
  $xiaoming->setname("王小明");
  $xiaoming->setgender("男");
  $lili = new guests;
  $lili->setname("李莉莉");
  $lili->setgender("女");
  echo $xiaoming->getname()."\t".$xiaoming->getgender()."<br />";
  echo $lili->getname()."\t".$lili->getgender();
?>
```

运行结果如图 8-1 所示。



图 8-1　程序运行结果

【案例分析】：

（1）其中用 class 关键字声明一个类，而这个类的名称是 guests。在大括号内写入类的属性和方法。其中 private $name、private $gender 为类 guests 的自有属性，用 private 关键字声明，也就是说只有在类内部的方法可以访问它们，类外部是不能访问的。

（2）function setname($name)、function getname()、function setgender($gender)和 function getgender()就是类方法，它们可以对 private $name、private $gender 这两个属性进行操作。$this 是对类本身的引用。用“->”连接类属性，格式如$this->name 和$this->gender。

（3）之后用 new 关键字生成一个对象，格式为$object = new Classname;，它的对象名是$xiaoming。当程序通过 new 生成一个类 guests 的实例，也就是对象$xiaoming 的时候，对象$xiaoming 就拥有了类 guests 的所有属性和方法。然后就可以通过“接口”也就是这个对象的方法（也就是类的方法的拷贝）来对对象的属性进行操作。

（4）通过接口 setname($name)给实例$xiaoming 的属性$name 赋值为 XiaoMing，通过 setgender($gender)给实例$xiaoming 的属性$gender 赋值为 male。同样的道理，通过接口操作了实例$lili 的属性。最后通过接口 getname()、getgender()返回不同的两个实例的属性$name 和$gender，并且打印出来。

8.3　构造方法和析构方法

构造方法存在于每个声明的类中，主要作用是执行一些初始化的任务。如果类中没有直接声明构造方法，那么类会默认生成一个没有参数且内存为空的构造方法。

在 PHP 中，声明构造方法的方法有两种，在 PHP 5 版本之前，构造方法的名称必须与类名相同；从 PHP 5 版本开始，构造方法的名称必须以两个下划线开头，即“--xonstruct”。具体的语法格式如下：

```
Function--construct([mixed args]){
            //方法的内容
}
```

一个类只能声明一个构造方法。构造方法中的参数是可选的，如果没有传入参数，那么将使用默认参数为成员变量进行初始化。

在例 8.1 中，对实例$xiaoming 的属性$name 进行赋值，还需要通过使用接口 setname($name)进行操作，如$xiaoming->setname("XiaoMing")。 如果想在生成实例$xiaoming 的同时就对此实例

的属性$name 进行赋值，该怎么办呢？

这时就需要构造方法“_construct()”了。这个函数的特性是，当通过关键字 new 生成实例的时候，它就会被调用执行。它的用途就是经常对一些属性进行初始化，也就是给一些属性进行初始化的赋值。

下面通过实例介绍构造方法的使用方法和技巧。

【例 8.2】（实例文件：ch08\8.2.php）

```
<?php
 class guests{                              //定义类 guests
    private $name;
    private $gender;
    function __construct($name,$gender){  //定义函数 function __construct
         $this->name = $name;
     $this->gender = $gender;
    }
    function getname(){                      //定义函数 getname
          return $this->name;
    }
    function getgender(){                     //定义函数 getgender
          return $this->gender;
    }
 };
 $xiaoming = new guests("赵大勇","男");
 $lili =  new guests("方芳芳","女");
 echo $xiaoming->getname()."\t".$xiaoming->getgender()."<br />";
 echo $lili->getname()."\t".$lili->getgender();
?>
```

运行结果如图 8-2 所示。

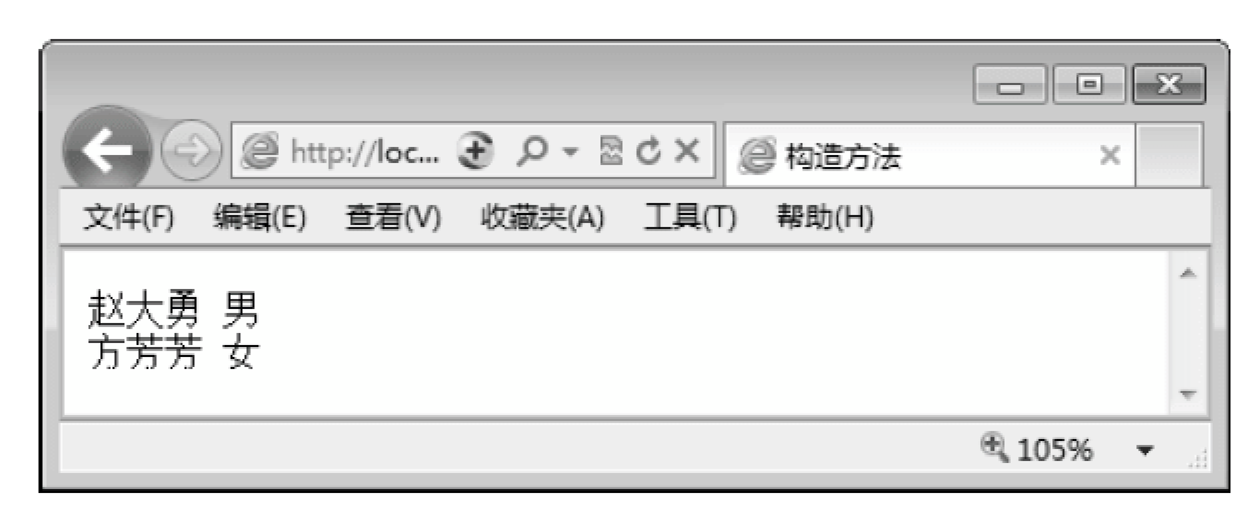


图 8-2　程序运行结果

要记住的是，构造方法是不能返回（return）值的。

有构造方法，就有它的反面“析构方法”（desturctor）。它是在对象被销毁的时候被调用执行的。但是因为 PHP 在每个请求的最终都会把所有资源释放，所以析构方法的意义是有限的。具

体使用的语法格式如下：

```
function__destruct(){
        //方法的内容，通常完成一些在对象销毁前的清理任务
}
```

由于 PHP 具有垃圾回收机制，可以自动清除不再使用的对象，从而释放更多的内存。析构方法在垃圾回收程序执行前被调用的方法，是 PHP 编程中的可选内容。

不过，在执行某些特定行为时，还是有用的，比如在对象被销毁时清空资源或者记录日志信息。

以下两种情况下，析构方法可能被调用执行。

- 代码运行时，当所有对于某个对象的 reference（引用）被毁掉的情况下；
- 当代码执行到最后，并且 PHP 停止请求的时候，调用 destructor 函数。

8.4 访问方法

另一个很好用的函数是访问方法（accessor）。由于 OOP 思想并不鼓励直接从类的外部访问类的属性，以强调封装性，所以可以使用_get 和_set 方法来达到此目的，也就是说要使用访问函数。无论何时，类属性被访问和操作，访问方法都会被激发。通过使用它们，可以避免直接对类属性的访问。

下面通过实例介绍访问方法的使用方法和技巧。

【例 8.3】（实例文件：ch08\8.3.php）

```
<?php
 class guests{
   public $property;
   function _set($propName,$propValue){
       $this->$propName = $propValue;
   }
   function _get($propName){
        return $this->$propName;
   }
 };
 $xiaoshuai = new guests;
 $xiaoshuai->name = "刘小帅";
 $xiaoshuai->gender = "男性";
 $dingdang = new guests;
 $dingdang->name = "丁叮当";
 $dingdang->gender = "女性";
 $dingdang->age = 28;
```

```
    echo $xiaoshuai->name." 是 ".$xiaoshuai->gender."<br />";
    echo    $dingdang->name."    是    一    位        ".$dingdang->age."    岁
".$dingdang->gender."<br />";
  ?>
```

运行结果如图 8-3 所示。

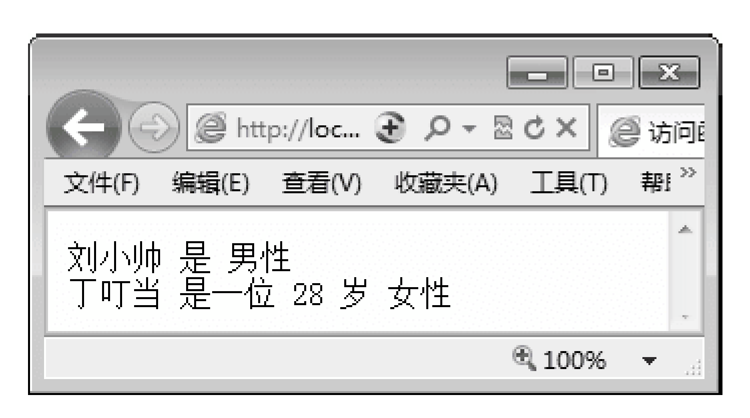


图 8-3　程序运行结果

【案例分析】：

（1）$xiaoshuai 为类 guest 的实例。直接添加属性 name 和 gender，并且赋值，如"$xiaoshuai->name = "刘小帅"; $xiaoshuai->gender = "男性";"，此时，类 guest 中的_set 函数被调用。$dingdang 实例为同样的过程。另外，$dingdang 实例添加了一个对象属性 age。

（2）echo 命令中用到的对象属性，如$xiaoshuai->name 等，则是调用了类 guest 中的_get 函数。

（3）此例中，_set 方法的格式为：

```
    function _set($propName,$propValue){
         $this->$propName = $propValue;
    }
```

_get 方法的格式为：

```
    function _get($propName){
          return $this->$propName;
    }
```

其中，$propName 为“属性名”，$propValue 为“属性值”。

8.5　类的继承

继承（inheritance）是 OOP 中最重要的特性与概念。父类拥有其子类的公共属性和方法。子类除了拥有父类具有的公共属性和方法以外，还拥有自己独有的属性和方法。

PHP 使用关键字 extends 来确认子类和父类，实现子类对父类的继承。具体的语法格式如下：

```
  class 子类名称 extends 父类名称{
       //子类成员变量列表
       function 成员方法(){              //子类成员方法
            //方法内容
       }
```

```
}
```

下面通过实例介绍类的继承方法。

【例 8.4】（实例文件：ch08\8.4.php）

```
<?php
  class Vegetables{
     var $tomato ="西红柿";                          //定义变量
     var $cucumber="黄瓜";
  };
  class VegetablesType extends Vegetables{               //类之间的继承
     var $potato="马铃薯";                          //定义子类的变量
     var $radish="萝卜";
  };
  $vegetables=new VegetablesType();                    //实例化对象
  echo "蔬菜包括：".$vegetables->tomato."，".$vegetables->t cucumber."，".$vegetables-> potato."，".$vegetables-> radish;
?>
```

运行结果如图 8-4 所示。

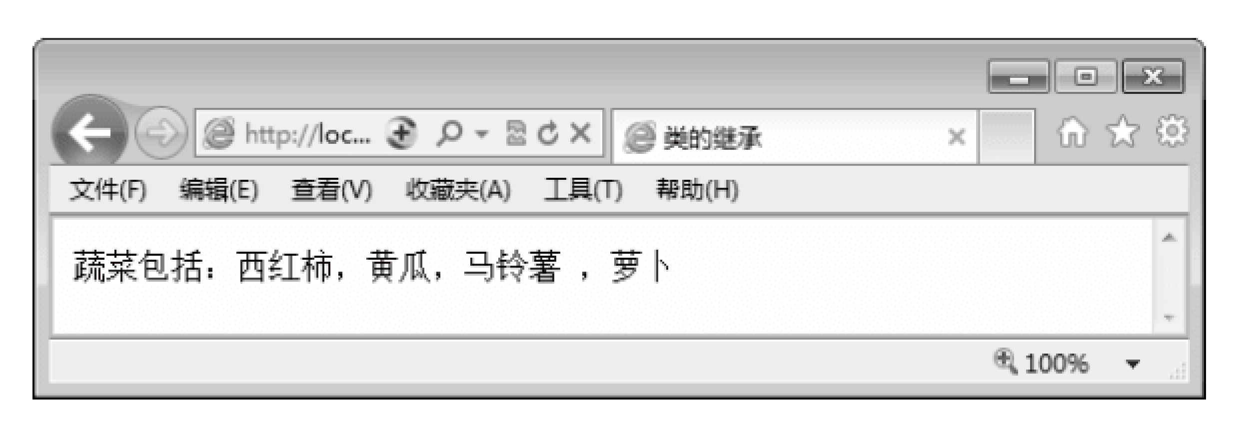


图 8-4　程序运行结果

从结果可以看出，本实例创建了一个蔬菜父类，子类通过关键字 extends 继承了蔬菜父类中的成员属性，最后对子类进行实例化操作。

8.6　高级特性

本节将学习 PHP 中关于类的一些高级特性。

8.6.1　静态属性和方法

声明类属性或方法为 static（静态），就可以不实例化类而直接访问。静态属性不能通过一个类已实例化的对象来访问（但静态方法可以）。由于静态方法不需要通过对象即可调用，所以伪变量 $this 在静态方法中不可用。静态属性不可以由对象通过->操作符来访问。自 PHP 5.3.0 起，可以用一个变量来动态调用类。但该变量的值不能为关键字 self，parent 或 static。

静态属性不需要实例化就可以直接使用，调用格式为“类名：：静态属性名”。同样，静态方法也不需要实例化即可直接使用，调用格式为“类名：：静态方法名”。

【例 8.5】（实例文件：ch08\8.5.php）

```
<?php
class Gushi {
  public static $my_static = '洛阳亲友如相问，一片冰心在玉壶。<br/>';
  public function staticValue() {
     return self::$my_static;
  }
}
print Gushi::$my_static;
$gushi = new Gushi();
print $gushi ->staticValue();
?>
```

运行结果如图 8-5 所示。



图 8-5　程序运行结果

8.6.2　final 类和方法

从 PHP 5 开始，新增了一个 final 关键字。如果父类中的方法被声明为 final，则子类无法覆盖该方法。如果一个类被声明为 final，则不能被继承。

1. fina 方法不能被重写

如果希望类中的某个方法不能被子类重写，可以将设置该方法为 final 方法，只需要在方法前加上 final 修饰符。如果这个方法被子类重写，将会出现错误。

【例 8.6】（实例文件：ch08\8.6.php）

```
<?php
class Math{
     //计算两个数值的和
     public final function Sum($a,$b){
          return $a+$b;
     }


}
```

```
class M extends Math {
    public function Sum($a,$b) {  //重写 Sum 方法
        echo '这里先测试一下';
    }
}
$math = new M();
echo $math->Sum(10,20);
?>
```

运行结果如图 8-6 所示。从结果可以看出，final 方法不能被重写，否则会被报错。

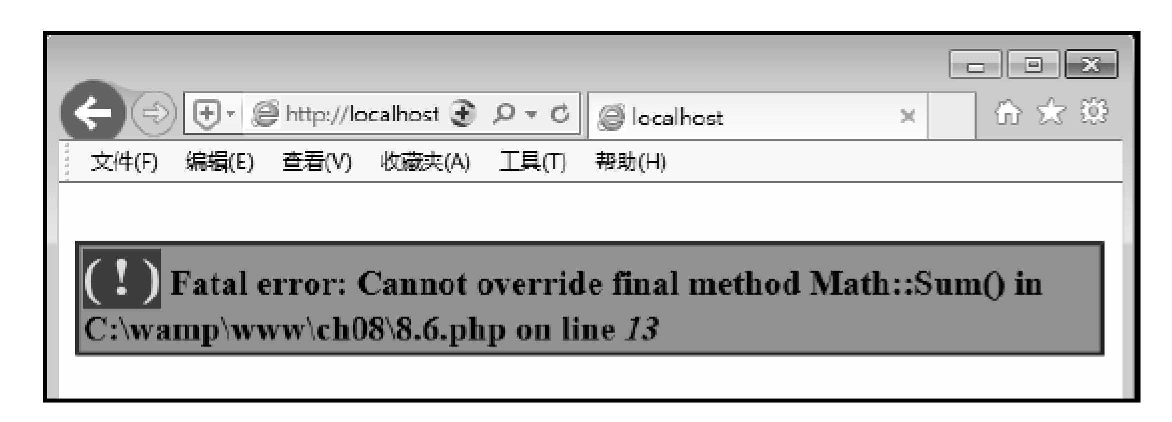


图 8-6　程序运行结果

2. final 类不能被继承

final 关键词可以终止类的继承，final 类不能有子类，final 方法不能被继承。

【例 8.7】（实例文件：ch08\8.7.php）

```
<?php
final class Poth{
    public $aa = 9.99;
}
$poth = new Poth();
echo $poth;
//声明 M 类，它继承自 Poth 类，但执行时会出错，final 类不能被继承。
class M extends Poth {


}
?>
```

运行结果如图 8-7 所示。



图 8-7　程序运行结果

8.7 抽象类和接口

抽象类和接口都是特殊的类，因为它们都不能被实例化。本章主要讲述两者的使用方法和技巧。

8.7.1 抽象类

抽象类只能作为父类使用，因为抽象类不能被实例化。抽象类使用关键字 abstract 来声明，具体的使用语法格式如下：

```
abstract class 抽象类名称{
     //抽象类的成员变量列表
     abstract function 成员方法 1(参数);              //抽象类的成员方法
     abstract function 成员方法 2(参数);              //抽象类的成员方法
 }
```

抽象类和普通类的主要区别在于抽象类的方法没有方法内容，而且至少包含一个抽象方法。另外抽象方法也必须使用关键字 abstract 来修饰，抽象方法后必须有分号。

【例 8.8】（实例文件：ch08\8.8.php）

```
<?php
    abstract class MyObject{                          //定义抽象类
        abstract function service($getName,$price,$num);
    }
    class MyBook extends MyObject{
        function service($getName,$price,$num){
            echo '购买的商品是'.$getName.'，商品的价格是：'.$price.' 元。';
            echo '您购买的数量为：'.$num.' 本。';
        }
    }
    class MyComputer extends MyObject{                 //继承抽象类
        function service($getName,$price,$num){
            echo '您购买的商品是'.$getName.'，该商品的价格是：'.$price.' 元。';
            echo '您购买的数量为：'.$num.' 本。';
        }
    }
    $book = new MyBook();
    $computer = new MyComputer();
    $book -> service('《PHP 7 从入门到精通》',59,15);
    echo '<p>';
    $computer -> service('MySQL5.7 从零开始学',65,10);
?>
```

运行结果如图 8-8 所示。

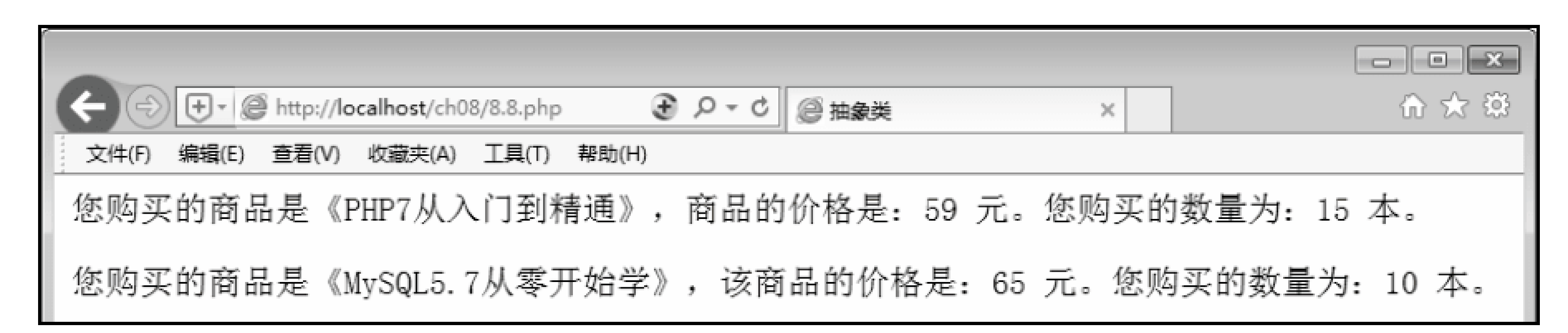


图 8-8　程序运行结果

8.7.2　接口

继承特性简化了对象和类的创建，增加了代码的可重用性。但是 PHP 只支持单继承，如果想实现多继承，就需要使用接口。PHP 可以实现多个接口。

接口类通过关键字 interface 来声明，接口中不能声明变量，只能使用关键字 const 声明为常量的成员属性，接口中声明的方法必须是抽象方法，并且接口中所有的成员都必须具有 public 访问权限。具体的使用语法格式如下：

```
interface 接口名称{                              //使用 interface 关键字声明接口
      //常量成员                              //接口中成员只能是常量
      //抽象方法                              //成员方法必须是抽象方法
 }
```

与继承使用 extends 关键字不同的是，实现接口使用的是 implement 关键字：

```
class 接口类 implement 接口名称{}
```

实现接口的类必须实现接口中声明的所有方法，除非这个类被声明为抽象类。

【例 8.9】（实例文件：ch08\8.9.php）

```
<?php
 interface Maxmin{
     //这两个方法必须在子类中继承,修饰符必须为public
    public function getMax();
    public function getMin();
}
class msm implements Maxmin {
    private $aa = 33;
    private $bb = 66;
    //具体实现接口声明的方法
    public function getMax(){
        return $this->bb;
    }
    public function getMin(){
        return $this->aa;
```

```
        }
        //这里还可以有自己的方法
        public function getOther(){
            return '这里是自己的方法';
        }
    }
    $msm = new msm();
    echo $msm->getMax();
    echo '<br>';
    echo $msm->getMin();
    echo '<br>';
    echo $msm->getOther();
    ?>
```

运行结果如图 8-9 所示。



图 8-9　程序运行结果

通过上述实例，可以总结如下的要点：

- 在 PHP 中类的继承只能是单独继承，即由一个父类（基类）继承，而且可以一直继承下去。PHP 不支持多方继承，即不能由一个以上的父类进行继承，即类 C 不能同时继承类 A 和类 B。
- 由于 PHP 支持多方继承，为了对特定类的功能进行拓展，就可以使用接口（interface）来实现类似于多方继承的好处。接口用 interface 关键字声明，并且单独设置接口方法。
- 一个类可以继承于一个父类，同时使用一个或多个接口。类还可以直接继承于某个特定的接口。
- 类、类的属性和方法的访问，都可以通过访问修饰符进行控制。访问修饰符放在属性和类的前面则表示 public 为公共属性或方法，private 为自有属性或方法，protected 为可继承属性或方法。
- 关键字 final 放在特定的类前面，表示此类不能再被继承。final 放在某个类方法前面，表示此方法不能在继承后被“覆写”或重新定义。

8.8　面向对象的多态性

多态性是指同一操作作用于不同的类的实例，将产生不同的执行结果，即不同类的对象收到相同的消息时，得到不同的结果。在 PHP 中，实现多态的方法有两种，包括通过继承实现多态和通过接口实现多态。

8.8.1　通过继承实现多态

通过继承可以实现多态的效果，下面通过一个实例来理解实现多态的方法。

【例 8.10】（实例文件：ch08\8.10.php）

```
<?php
  abstract class Vegetables{                          //定义抽象类 Vegetables
     abstract function go_Vegetables();               //定义抽象方法 go_Vegetables
   }
class Vegetables_potato extends Vegetables{           //马铃薯类继承蔬菜类
      public function go_Vegetables(){                //重写抽象方法
           echo "我们开始种植马铃薯" ;                  //输出信息
      }
    }
   class Vegetables_radish extends Vegetables{        //萝卜类继承蔬菜类
      public function go_Vegetables(){                //重写抽象方法
           echo "我们开始种植萝卜" ;
      }
    }
   function change($obj){                         //自定义方法根据对象调用不同的方法
       if($obj instanceof Vegetables ){
           $obj->go_Vegetables();
       }else{
            echo "传入的参数不是一个对象";              //输出信息
       }
    }
    echo "实例化 Vegetables_potato: ";
    change(new Vegetables_potato());                  //实例化 Vegetables_potato
    echo "<br />";
    echo "实例化 Vegetables_ radish: ";
    change(new Vegetables_ radish ());                //实例化 Vegetables_ radish
?>
```

运行结果如图 8-10 所示。



图 8-10　程序运行结果

从结果可以看出，本实例创建了一个抽象类 Vegetables，用于表示各种蔬菜的种植方法，然后

让子类继承这个 Vegetables。

8.8.2 通过接口实现多态

下面通过接口的方式，实现和上面实例一样的效果。

【例 8.11】（实例文件：ch08\8.11.php）

```
<?php
  interface Vegetables{                              //定义接口 Vegetables
     public function go_Vegetables();                //定义接口方法
   }
 class  Vegetables_potato  implements  Vegetables{                         //
Vegetables_potato 实现 Vegetables 接口
      public function go_Vegetables(){               //定义 go_Vegetables 方法
           echo "我们开始种植马铃薯" ;                  //输出信息
      }
    }
    class  Vegetables_radish  implements  Vegetables{                      //
Vegetables_radish 实现 Vegetables 接口
      public function go_Vegetables(){               //定义 go_Vegetables 方法
           echo "我们开始种植萝卜" ;                    //输出信息
      }
    }
    function change($obj){                         //自定义方法根据对象调用不同的方法
       if($obj instanceof Vegetables ){
           $obj->go_Vegetables();
       }else{
            echo "传入的参数不是一个对象";                //输出信息
       }
    }
     echo "实例化 Vegetables_potato: ";
     change(new  Vegetables_potato());                                  // 实 例 化
Vegetables_potato
     echo "<br />";
     echo "实例化 Vegetables_ radish: ";
     change(new Vegetables_ radish ());                         //实例化 Vegetables_
radish
 ?>
 </body>
 </html>
```

运行结果如图 8-11 所示。



图 8-11 程序运行结果

从结果可以看出，本实例创建了一个接口 Vegetables，然后定义一个空方法 go_Vegetables()，接着定义 Vegetables_potato 和 Vegetables_radish 子类承接接口 Vegetables。最后通过 instanceof 关键字检查对象是否属于接口 Vegetables。

8.9 高手私房菜

技巧 1： 理解"（a < b ）? a ： b;"的含义

这是条件控制语句，是 if 语句的单行表示方法。它的具体格式是：

(条件判断语句)? 判断为 true 的行为 ：判断为 false 的行为；

if 语句的单行表示方式的好处是，可以直接对条件判断结果的返回值进行处理。比如可以直接把返回值赋值给变量。对于$varible = (a < b)? a : b;，如果 a<b 的结果为 ture，则此语句返回 a，并且直接赋值给$varible，如果 a<b 的结果为 false，则此语句返回 b，并且直接赋值给$varible。

这种表示方法可以节约代码的输入量，更重要的是可以提高代码执行的效率。由于 PHP 代码执行是对代码由上至下的一个过程，所以代码的行数越少，越能节约代码读取的时间。在一行语句中就能对情况做出判断，并且对代码返回值进行处理，这无疑是一种效率相当高的代码组织方式。

技巧 2：说明抽象类和类的不同之处

抽象类是类的一种，通过在类的前面增加关键字 abstract 来表示。抽象类是仅仅用来继承的类。通过 abstact 关键字声明，就是告诉 PHP，这个类不再用于生成类的实例，仅仅是用来被其子类继承。可以说抽象类只关注于类的继承。抽象方法就是在方法前面添加关键字 abstract 声明的方法。抽象类中可以包含抽象方法。一个类中只要有一个方法通过关键字 abstract 声明为抽象方法，则整个类都要声明为抽象类。然而，特定的某个类即便不包含抽象方法，也可以通过 abstract 声明为抽象类。

8.10 经典习题

（1）制作一个输出当前时间戳的例子。

（2）制作一个输出当前时间和日期的例子。

（3）制作一个包含比较两个时间大小的例子。

（4）制作一个实现倒计时功能的例子。

第 9 章　错误处理和异常处理

当 PHP 代码运行时，会发生各种错误：可能是语法错误，通常是程序员造成的编码错误或错别字；可能是拼写错误或语言中缺少的功能（可能由于浏览器差异）；可能是由于来自服务器或用户的错误输出而导致的错误。当然，也可能是由于许多其他不可预知的因素。本章主要讲述错误处理和异常处理。

本章学习目标

- 了解常见的错误和异常
- 掌握处理错误的方法
- 掌握处理异常的方法

9.1　常见的错误和异常

错误和异常是编程中经常出现的问题。本节将主要介绍常见的错误和异常。

1. 拼写错误

拼写代码时要求程序员要非常仔细，并且还需要认真地检查编写完成的代码，否则会出现不少编写上的错误。

另外 PHP 中常量和变量都是区分大小写的，例如把变量名 abc 写成 ABC，都会出现语法错误。PHP 中的函数名、方法名、类名不区分大小写，但建议使用与定义时相同的名字。魔术常量不区分大小写，但是建议全部大写，包括_LINE_、_FILE_、_DIR_、_FUNCTION_、_CLASS_、_METHOD_、_NAMESPACE_。知道这些规则，就可以避免大小写的错误。

另外，编写代码时有时需要输入中文字符，编程人员容易在输完中文字符后忘记切换输入法，从而导致输入的小括号、分号或者引号等出现错误，当然，这种错误输入在大多数编程软件中显示的颜色会跟正确的输入显示的颜色不一样，容易发现，但还是应该细心以减少错误的出现。

2. 单引号和双引号的混用

单引号、双引号在 PHP 中没有特殊的区别，都可以用来创建字符串。但是必须使用同一种单或双引号来定义字符串，如 'Hello" 和 "Hello' 为非法的字符串定义。单引号串和双引号串在 PHP 中的处理是不相同的。双引号串中的内容可以被解释而且替换，而单引号串中的内容总被认为是普通字符。

另外，缺少单引号或者双引号也是经常出现的问题。例如：

```
echo "错误处理的方法;
```

其中缺少双引号，运行时会提示错误。

3. 括号使用混乱

首先需要说明的是，在 PHP 中，括号包含两种语义，可以是分隔符也可以是表达式。例如：

- 分隔符的作用比较常用，比如（1+4）*4 等于 20。
- 在 (function(){})();中，function 之前的一对括号作为分隔符，后面的括号表示立即执行这个方法。

另外由于括号的使用层次比较多，所以可能会导致括号不匹配的错误，如以下代码所示：

```
If((($a==$b)and($b==$c))and($c==$d){        //此处缺少一个括号
     echo "正确的括号使用方法！"
}
```

4. 等号与赋值符号混淆

等号与赋值符号混淆的错误一般出现在 if 语句中，而且这种错误在 PHP 中不会产生错误信息，所以在查找错误时往往不容易被发现。比如：

```
if(s =1)
  echo ("没有找到相关信息");
```

上面的代码在逻辑上是没有问题的，它的运行结果是将 1 赋值给了 s，如果成功则弹出对话框，而不是对 s 和 1 进行比较，这不符合开发者的本意。

5. 缺少美元符号

在 PHP 中，设置变量时需要使用美元符号“$”，如果不添加美元符号就会引起解析错误。如以下代码所示：

```
for($s =1;$s<=10;s++){                  //缺少一个变量的美元符号
  echo ("缺少美元符号！");
}
```

需要修改 s++为$s++即可。如果$s<=10;缺少美元符号，则会进入无限循环状态。

6. 调用不存在的常量和变量

如果调用没有声明的常量或变量，将会触发 NOTICE 错误。例如下面的代码中，输出时错误书写了变量的名称。

```
<?php
  $abab= "错误处理的方法"                 //缺少一个变量的美元符号
  echo $abba;
?>
```

如果运行程序，会出现如图 9-1 所示的错误。

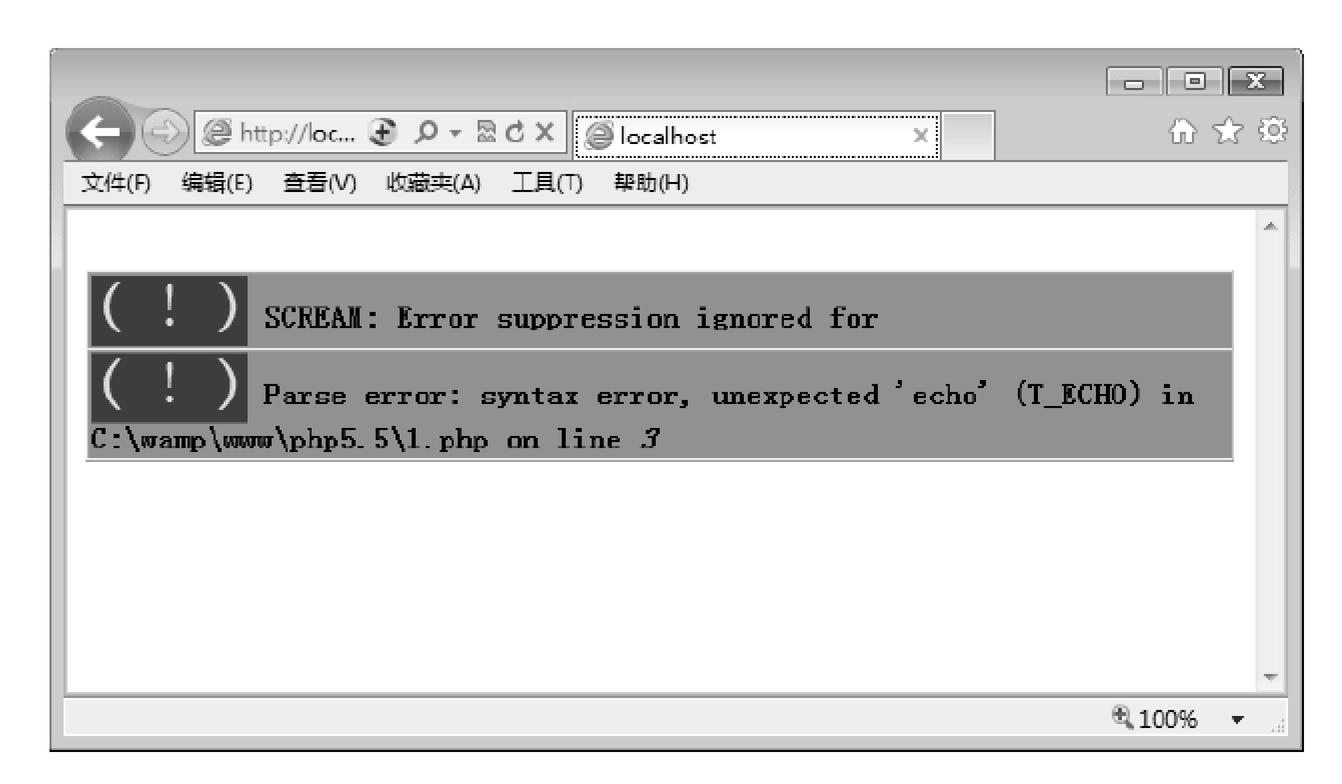


图 9-1　调用不存在的变量

7. 调用不存在的文件

如果调用不存在的文件，程序将会停止运行。例如下面的代码：

```
<?php
  include("mybook.txt");                    //调用一个不存在的文件
?>
```

运行后将会弹出如图 9-2 所示的错误提示信息。

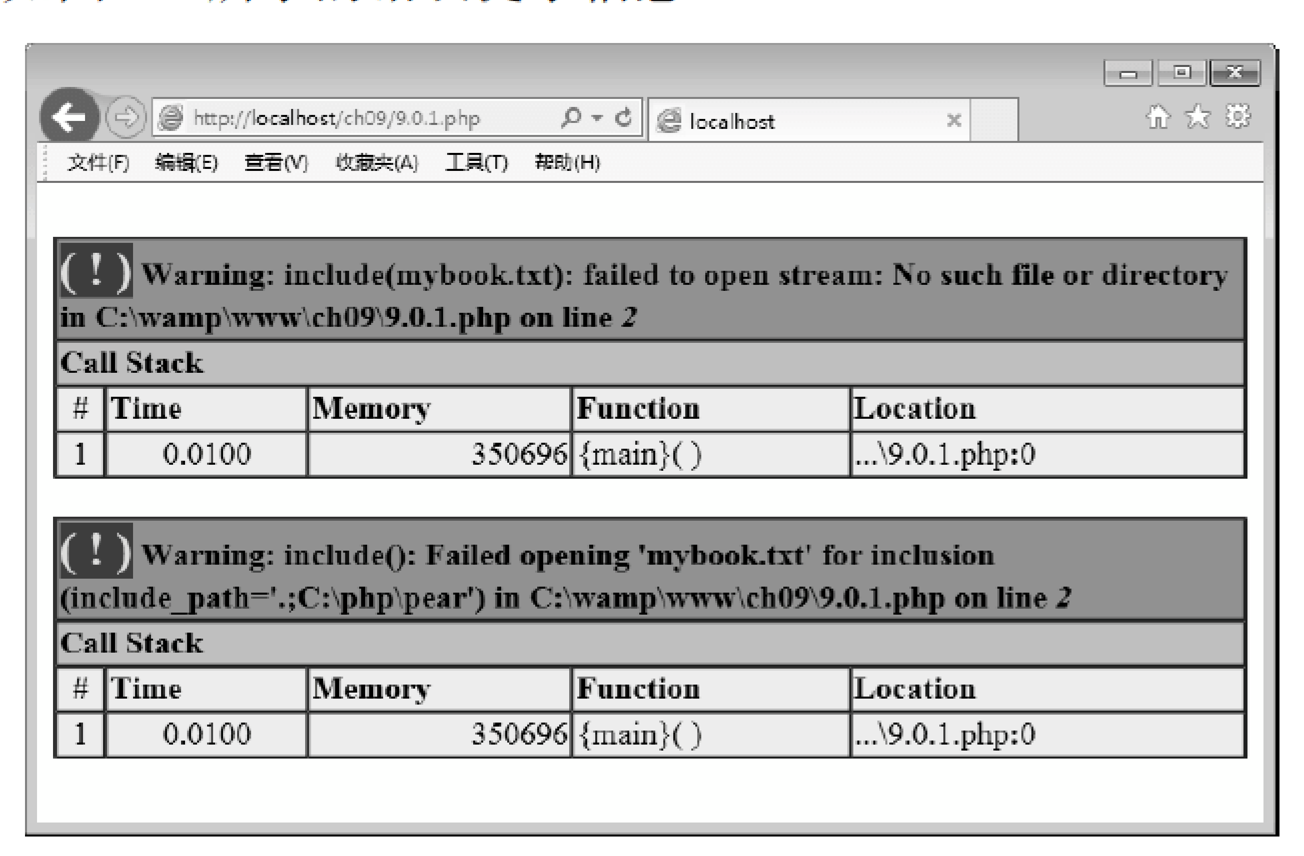


图 9-2　调用不存在的文件

8. 环境配置的错误

如果环境配置不当，也会给运行带来错误，例如操作系统、PHP 配置文件和 PHP 的版本等，如果配置不正确，将会提示文件无法打开、操作权限不具备和服务器无法连接等错误信息。

首先不同的操作系统采用不同的路径格式，这些都会导致程序运行错误。另外，PHP 在不同的操作系统上的功能也会有差异，数据库的运行也会在不同的操作系统中有问题出现等。另外 PHP 的配置也很重要，由于各个计算机的配置方法不尽相同，当程序的运行环境发生变化时，也会出现这样或那样的问题。最后是 PHP 的版本问题，PHP 的高版本在一定程度上可以兼容低版本，但是高版本编写的程序到低版本中运行，会出现意想不到的问题，这些都是有关环境配置的不同而引起的错误。

9. 数据库服务器连接错误

由于 PHP 应用于动态网站的开发，所以经常会对数据库进行基本的操作，在操作数据库之前，需要连接数据库服务，如果用户名或者密码设置不正确，或者数据库不存在，或者数据库的属性不允许访问等，都会在程序运行中出现错误。

例如以下的代码，在连接数据库的过程中，密码编写是错误的。

```
<?php
  $conn=mysqli_connect("localhost","root","root");          //连接 MySQL
服务器
?>
```

运行后将会弹出如图 9-3 所示的错误提示信息。

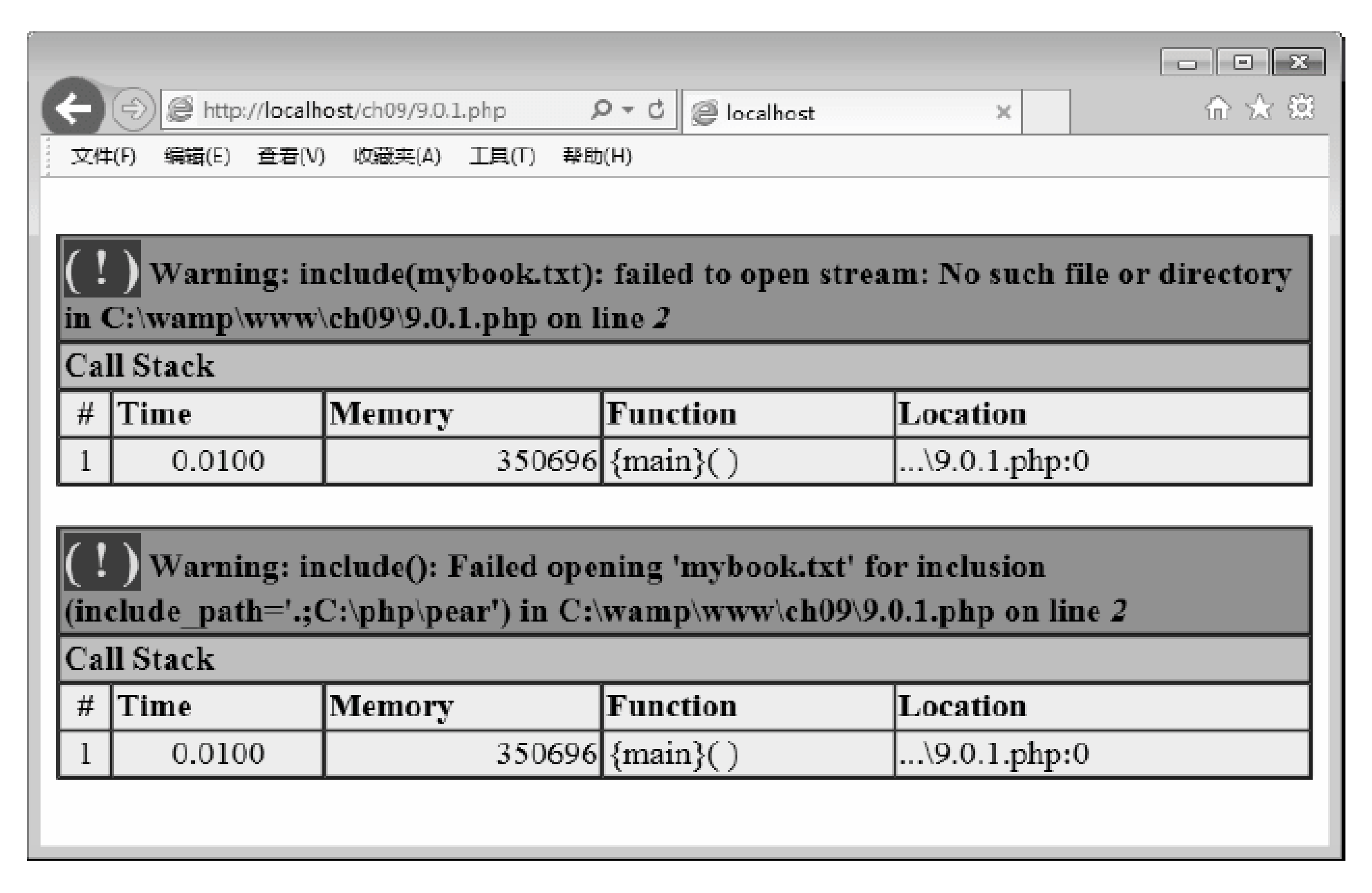


图 9-3 无法连接数据库

9.2 错误处理

常见的错误处理方法包括使用错误处理机制、使用 DIE 语句调试、自定义错误和错误触发器等。本节将讲述如何处理程序中的错误。

9.2.1 php.ini 中的错误处理机制

在前面的实例中，错误提示会显示错误的信息，如错误文件的行号等，这是 PHP 最基本的错误报告机制。php.ini 文件规定了错误的显示方式，包括配置选项的名称、默认值和表述的含义等。常见的错误配置选项的内容如表 9-1 所示。

表 9-1　常见的 php.ini 文件中控制错误显示的配置选项含义

名称	默认值	含义
display_errors	On	设置错误作为 PHP 的一部分输出。开发的过程中可以采用默认的设置，但是为了安全考虑，在生产环境中还是设置为 Off 比较好
error_reporting	E_all	这个设置会显示所有的出错信息。这种设置会让一些无害的提示也显示，所以可以设置 error_reporting 的默认值为 error_reporting = E_ALL & ~E_NOTICE，这样只会显示错误和不良编码
error_log	null	设置记录错误日志的文件。默认情况下将错误发送到 Web 服务器日志，用户也可以指定写入的文件
html_errors	On	控制是否在错误信息中采用 HTML 格式
log_errors	Off	控制是否应该将错误发送到主机服务器的日志文件
display_startup_errors	Off	控制是否显示 PHP 启动时的错误
track_errors	Off	设置是否保存最近一个警告或错误信息

9.2.2　应用 DIE 语句调试

使用 DIE 语句调试的优势是，不仅可以显示错误的位置，还可以输出错误信息。一旦出现错误，程序将会终止运行，并在浏览器上显示出错之前的信息和错误信息。

在上一节中曾经介绍用不存在的文件会提示错误信息，如果运用 DIE 调试，将会输出自定义的错误信息。

【例 9.1】（实例文件：ch09\9.1.php）

```
<?php
  if(!file_exists("welcome.txt")){    //判断文件是否存在
  die("文件不存在");
   }
  else{
   $file=fopen("welcome.txt","r");
   }
?>
```

运行后结果如图 9-4 所示。



图 9-4　应用 DIE 语句调试

和基本的错误报告机制相比，使用 DIE 语句调试显得更有效，这是由于它采用了一个简单的错误处理机制，在错误之后终止了脚本。

9.2.3 自定义错误和错误触发器

简单地终止脚本并不是恰当的方式。本节将讲述如何自定义错误和错误触发器。创建一个自定义的错误处理器非常简单，用户可以创建一个专用函数，然后在 PHP 中发生错误时调用该函数。

自定义的错误函数的语法格式如下：

```
error_function(error_level,error_message,error_file,error_line,error_context)
```

该函数必须至少包含 error level 和 error message 下划线两个参数，另外 3 个参数，即 error_file、error_line 和 error_context 是可选的。各个参数的具体含义如表 9-2 所示。

表 9-2 各个参数的含义

参数	含义
error_level	必需参数。为用户定义的错误规定错误报告级别。必须是一个整数
error_message	必需参数。为用户定义的错误规定错误消息
error_file	可选参数。规定错误在其中发生的文件名
error_line	可选参数。规定错误发生的行号
error_context	可选参数。规定一个数组，包含当错误发生时在用的每个变量以及它们的值

参数 error_level 定义错误规定的报告级别，这些错误报告级别是错误处理程序旨在处理的错误的不同类型。具体的级别值和含义如表 9-3 所示。

表 9-3 错误的级别值和含义

数值	常量	含义
2	E_WARNING	非致命的 run-time 错误。不暂停脚本执行
8	E_NOTICE	Run-time 通知。脚本发现可能有错误发生，但也可能在脚本正常运行时发生
256	E_USER_ERROR	致命的用户生成的错误。这类似于程序员使用 PHP 函数 trigger_error() 设置的 E_ERROR
512	E_USER_WARNING	非致命的用户生成的警告。这类似于程序员使用 PHP 函数 trigger_error()设置的 E_WARNING
1024	E_USER_NOTICE	用户生成的通知。这类似于程序员使用 PHP 函数 trigger_error() 设置的 E_NOTICE
4096	E_RECOVERABLE_ERROR	可捕获的致命错误。类似 E_ERROR，但可被用户定义的处理程序捕获
8191	E_ALL	所有错误和警告

下面通过实例来讲解如何自定义错误和错误触发器。

首先创建一个处理错误的函数：

```
function customError($errno, $errstr)
 {
 echo "<b>错误:</b> [$errno] $errstr<br />";
 echo "终止程序";
 die();
}
```

上面的代码是一个简单的错误处理函数。当它被触发时，它会取得错误级别和错误消息。然后输出错误级别和消息，并终止程序。

创建了一个错误处理函数后，下面需要确定在何时触发该函数。在 PHP 中，使用 set_error_handler() 函数设置用户自定义的错误处理函数。该函数用于创建运行时的用户自己的错误处理方法。该函数会返回旧的错误处理程序，若失败，则返回 null。具体的语法格式如下：

```
set_error_handler(error_function,error_types)
```

其中，error_function 为必需参数，规定发生错误时运行的函数。error_types 是可选参数，如果不选择此参数，则表示默认值为 E_ALL。

在本例中，由于针对所有错误来使用自定义错误处理程序，具体的代码如下：

```
set_error_handler("customError");
```

下面通过尝试输出不存在的变量，来测试这个错误处理程序。

【例 9.2】（实例文件：ch09\9.2.php）

```
<?php
  //定义错误函数
  function customError($errno, $errstr){
  echo "<b>错误:</b> [$errno] $errstr";
  }
  //设置错误函数的处理
  set_error_handler("customError");
  //触发自定义错误函数
  echo($test);
?>
```

运行后结果如图 9-5 所示。

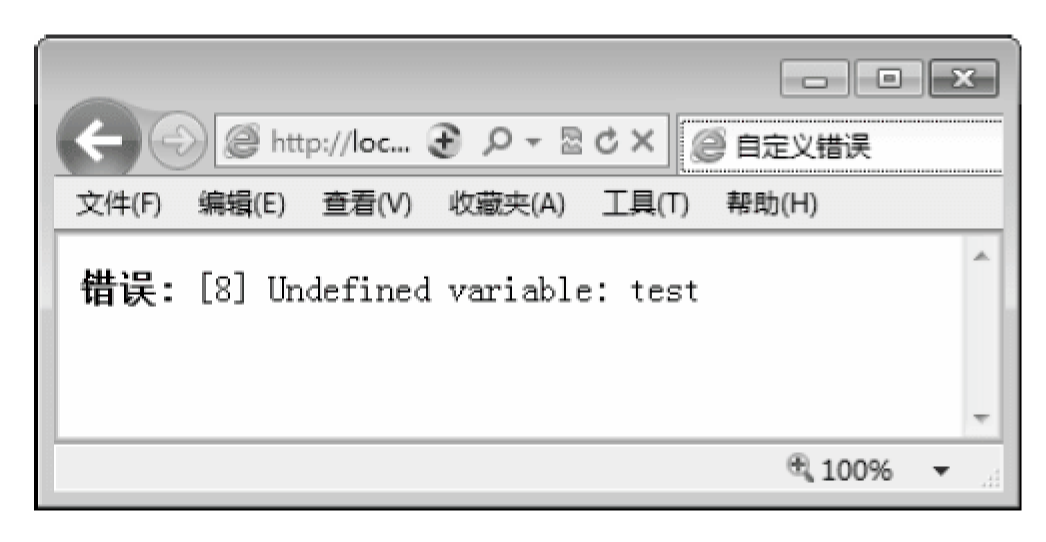


图 9-5　自定义错误

在脚本中用户输入数据的位置，当用户的输入无效时触发错误。在 PHP 中，这个任务由 trigger_error()完成。

trigger_error() 函数创建用户自定义的错误消息。trigger_error() 用于在用户指定的条件下触发一个错误消息。它与内建的错误处理器一同使用，也可以与由 set_error_handler() 函数创建的用户自定义函数一起使用。如果指定了一个不合法的错误类型，该函数返回 false，否则返回 true。

trigger_error() 函数的具体语法格式如下：

```
trigger_error(error_message,error_types)
```

其中 error_message 为必需参数，规定错误消息，长度限制为 1024 个字符。error_types 为可选参数，规定错误消息的错误类型，可能的值为 E_USER_ERROR、E_USER_WARNING 和 E_USER_NOTICE。

【例 9.3】（实例文件：ch09\9.3.php）

```
<?php
  $test=5;
  if ($test>4){
    trigger_error("Value must be 4 or below");  //创建自定义错误信息
  }
?>
```

运行后结果如图 9-6 所示。由于 test 数值为 5，将会则发生 E_USER_WARNING 错误。

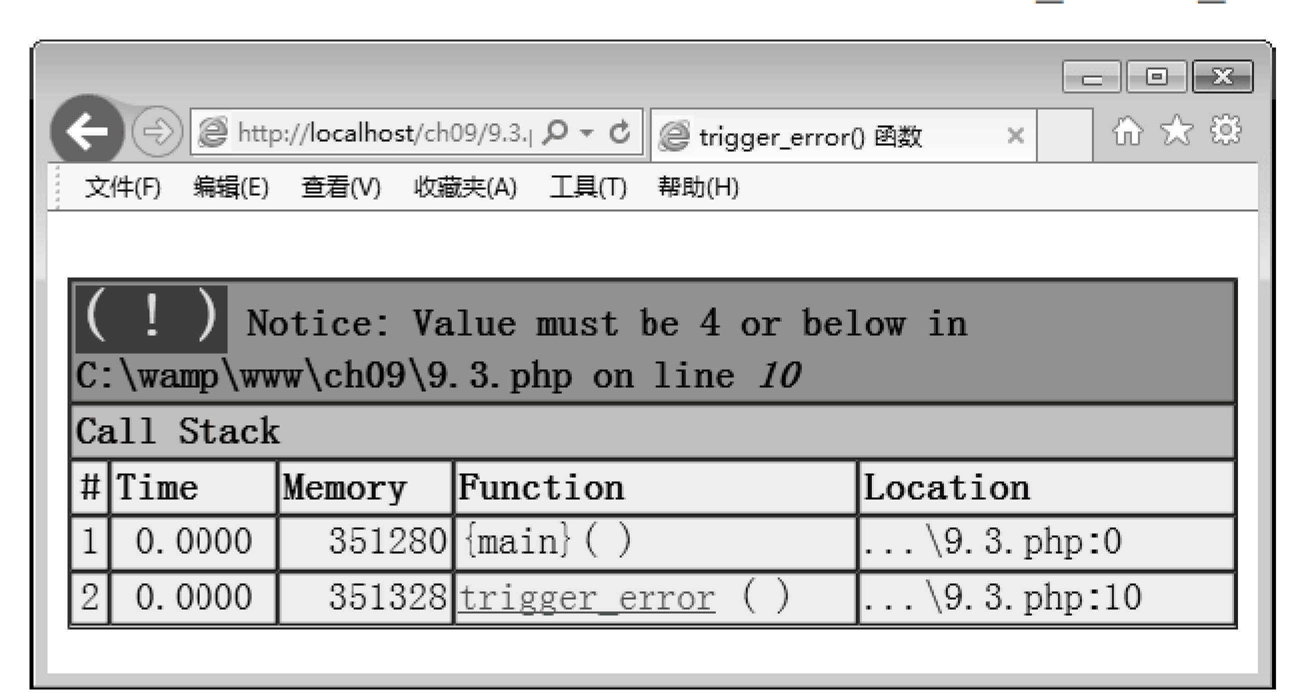


图 9-6　trigger_error() 函数

下面通过实例来讲述 trigger_error() 函数和自定义函数一起使用的处理方法。

【例 9.4】（实例文件：ch09\9.4.php）

```
<?php
  //定义错误函数
  function customError($errno, $errstr){
  echo "<b>错误:</b> [$errno] $errstr";
  }
  //设置错误函数的处理
  set_error_handler("customError",E_USER_WARNING);
  // trigger_error 函数
  $test=5;
  if ($test>4){
    trigger_error("Value must be 4 or below",E_USER_WARNING);
  }
?>
```

运行后结果如图 9-7 所示。



图 9-7　自定义函数和 trigger_error() 函数

9.2.4　错误记录

默认情况下，根据 php.ini 中的 error_log 配置，PHP 向服务器的错误记录系统或文件发送错误记录。通过使用 error_log() 函数，用户可以向指定的文件或远程目的地发送错误记录。

通过电子邮件向用户自己发送错误消息，是一种获得指定错误的通知的好办法。下面通过实例的方式来讲解。

【例 9.5】（实例文件：ch09\9.5.php）通过 email 发送错误信息

```
<?php
  //定义错误函数
  function customError($errno, $errstr){
  echo "<b>错误:</b> [$errno] $errstr <br/>";
  echo "错误记录已经发送完毕";
  error_log(" 错 误 : [$errno] $errstr",1, "someone@example.com", "From:
webmastere@example.com ");
 }
  //设置错误函数的处理
  set_error_handler("customError",E_USER_WARNING);
  // trigger_error 函数
  $test=5;
  if ($test>4){
    trigger_error("Value must be 4 or below",E_USER_WARNING);
  }
?>
```

运行后结果如图 9-8 所示。在指定的 someone@example.com 邮箱中将收到同样的错误信息。



图 9-8　error_log 函数

9.3 异常处理

异常（Exception）用于在指定的错误发生时改变脚本的正常流程。PHP 7 提供了一种新的面向对象的错误处理方法。本节主要讲述异常处理的方法和技巧。

9.3.1 异常的基本处理方法

异常处理用于在指定的错误（异常）情况发生时改变脚本的正常流程。当异常被触发时，通常会发生以下动作：

（1）当前代码状态被保存。

（2）代码执行被切换到预定义的异常处理器函数。

（3）根据情况，处理器也许会从保存的代码状态重新开始执行代码，终止脚本执行，或从代码中另外的位置继续执行脚本。

当异常被抛出时，其后的代码不会继续执行，PHP 会尝试查找匹配的 catch 代码块。如果异常没有被捕获，而且又没用使用 set_exception_handler()作相应的处理的话，那么将发生一个严重的错误，并且输出 Uncaught Exception（未捕获异常）的错误消息。

下面的实例抛出一个异常，同时不去捕获它。

【例 9.6】（实例文件：ch09\9.6.php）

```
<?php
  //创建带有异常的函数
  function checkNum($number){
   if($number>1){
     throw new Exception("Value must be 1 or below");
    }
   return true;
   }
  //抛出异常
  checkNum(2);
?>
```

运行后结果如图 9-9 所示。由于没有捕获异常，出现了下面的错误提示消息。

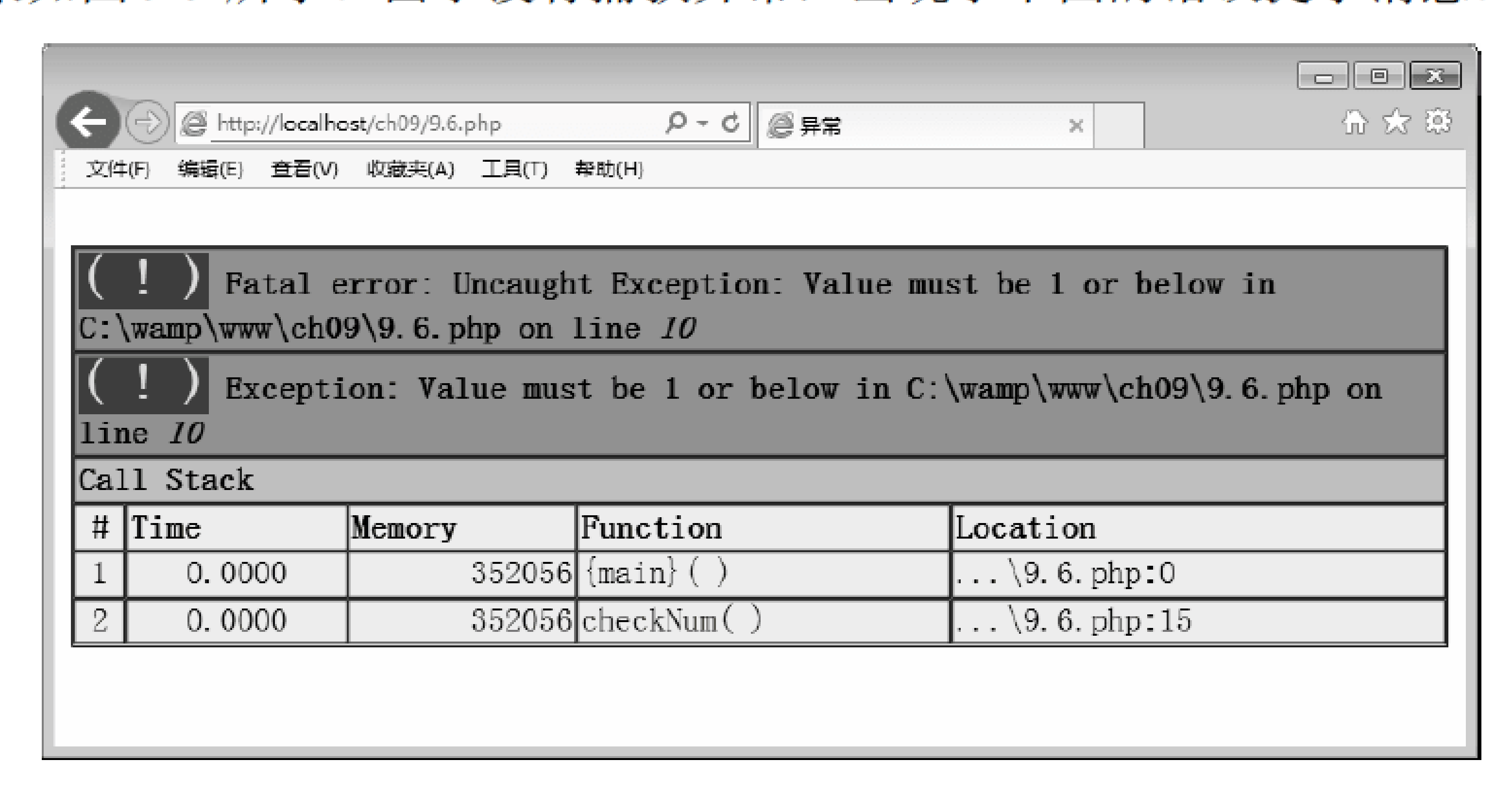


图 9-9 没有捕获异常

如果想避免出现上面的错误，需要创建适当的代码来处理异常。处理异常的程序应当包括如下几部分。

- try 代码块：使用异常的函数应该位于 try 代码块内。如果没有触发异常，则代码将照常继续执行。但是如果异常被触发，会抛出一个异常。
- throw 代码块：这里规定如何触发异常。每一个 throw 必须对应至少一个 catch。
- catch 代码块：catch 代码块会捕获异常，并创建一个包含异常信息的对象。

【例 9.7】（实例文件：ch09\9.7.php）

```
<?php
  //创建可抛出一个异常的函数
  function checkNum($number){
   if($number>1){
    throw new Exception("数值必须小于或等于 1");
    }
   return true;
   }
  //在 try 代码块中触发异常
  try{
   checkNum(2);
   //如果没有异常，则会显示以下信息
   echo '没有任何异常';
   }
  //捕获异常
  catch(Exception $e){
  echo '异常信息: ' .$e->getMessage();
 }
?>
```

运行后结果如图 9-10 所示。由于抛出异常后，捕获了异常，所以出现了下面的提示消息。

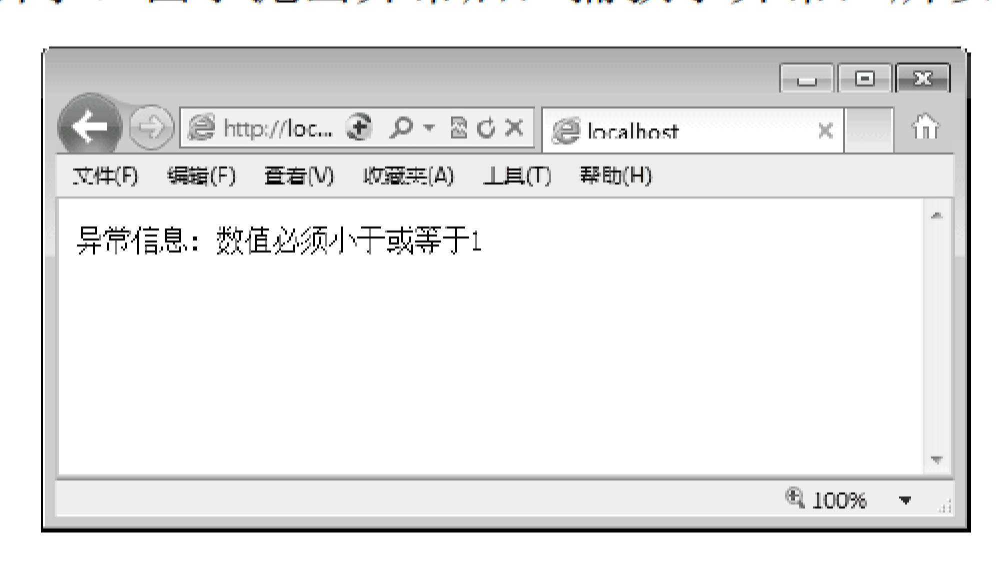


图 9-10　捕获异常

【案例分析】：

（1）首先创建 checkNum()函数。它检测数字是否大于 1。如果是，则抛出一个异常。

（2）在 try 代码块中调用 checkNum() 函数。
（3）checkNum() 函数中的异常被抛出。
（4）catch 代码块接收到该异常，并创建一个包含异常信息的对象 ($e)。
（5）通过从这个 exception 对象调用 $e->getMessage()，输出来自该异常的错误消息。

9.3.2 自定义的异常处理器

创建自定义的异常处理程序非常简单。只需要创建一个专门的类，当 PHP 中发生异常时，调用该类的函数即可。当然，该类必须是 exception 类的一个扩展。

这个自定义的 exception 类继承了 PHP 的 exception 类的所有属性，然后用户可向其添加自定义的函数。

下面通过实例讲解如何创建自定义的异常处理器。

【例 9.8】（实例文件：ch09\9.8.php）

```
<body>
<?php
  class customException extends Exception{
   public function errorMessage(){
    //错误消息
    $errorMsg = '异常发生的行： '.$this->getLine().' in '.$this->getFile()
    .': <b>'.$this->getMessage().'</b>不是一个有效的邮箱地址';
    return $errorMsg;
    }
   }
  $email = "someone@example.321com";
  try
   {
   //检查是否符合条件
   if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)  {
    //如果邮件地址无效，则抛出异常
    throw new customException($email);
    }
   }
  catch (customException $e){
   //显示自定义的消息
   echo $e->errorMessage();
   }
?>
```

运行后结果如图 9-11 所示。

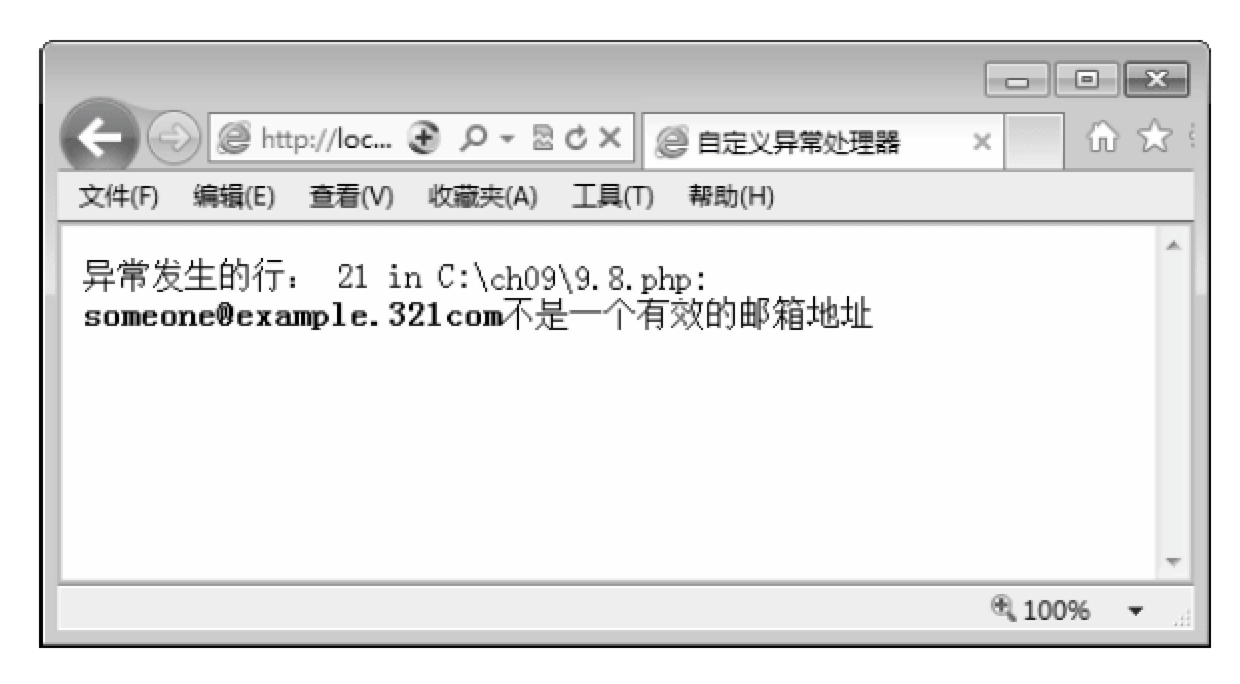


图 9-11　自定义异常处理器

【案例分析】：

（1）customException()类是作为旧的 exception 类的一个扩展而创建的。这样它就继承了旧类的所有属性和方法。

（2）创建 errorMessage()函数。如果 email 地址不合法，则该函数返回一条错误消息。

（3）把$email 变量设置为不合法的 email 地址字符串。

（4）执行 try 代码块，由于 email 地址不合法，因此抛出一个异常。

（5）catch 代码块捕获异常，并显示错误消息。

9.3.3　处理多个异常

在上面的实例中只是检查了邮箱地址是否有效。如果用户想检查邮箱是否为雅虎邮箱，或检查邮箱是否有效等，就会出现多个可能发生异常的情况。用户可以使用多个 if…else 代码块，或一个 switch 代码块，或者嵌套多个异常。这些异常能够使用不同的 exception 类，并返回不同的错误消息。

【例 9.9】（实例文件：ch09\9.9.php）

```
<?php
 class customException extends Exception{
 public function errorMessage(){
 //定义错误信息
 $errorMsg = '错误消息的行：'.$this->getLine().' in '.$this->getFile()
 .': <b>'.$this->getMessage().'</b> 不是一个有效的邮箱地址';
 return $errorMsg;
 }
 }
 $email = "someone@yahoo.com";
 try{
  //检查是否符合条件
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
   {
   //如果邮箱地址无效，则抛出异常
```

```
    throw new customException($email);
    }
   //检查邮箱是否是雅虎邮箱
   if(strpos($email, "yahoo") !== FALSE){
    throw new Exception("$email 是一个雅虎邮箱");
    }
   }
  catch (customException $e) {
   echo $e->errorMessage();
   }
  catch(Exception $e) {
   echo $e->getMessage();
   }
?>
```

运行后结果如图 9-12 所示。上面的代码测试了两种条件，如果任一条件不成立，则抛出一个异常。



图 9-12　处理多个异常

【案例分析】：

（1）customException()类是作为旧的 exception 类的一个扩展而创建的。这样它就继承了旧类的所有属性和方法。

（2）创建 errorMessage()函数。如果 email 地址不合法，则该函数返回一个错误消息。

（3）执行 try 代码块，在第一个条件下，不会抛出异常。

（4）由于 email 含有字符串 yahoo，第二个条件会触发异常。

（5）catch 代码块会捕获异常，并显示恰当的错误消息。

9.3.4　设置顶层异常处理器

所有未捕获的异常，都可以通过顶层异常处理器来处理。顶层异常处理器使用 set_exception_handler()函数来实现。

set_exception_handler()函数设置用户自定义的异常处理函数。该函数用于创建运行期间用户自

己的异常处理方法。该函数会返回旧的异常处理程序，若失败，则返回 null。具体的语法格式如下：

```
set_exception_handler(exception_function)
```

其中 exception_function 参数为必需参数，规定未捕获的异常发生时调用的函数，该函数必须在调用 set_exception_handler() 函数之前定义。这个异常处理函数需要一个参数，即抛出的 exception 对象。

【例 9.10】（实例文件：ch09\9.10.php）

```
<?php
  function myException($exception){        //定义顶层的异常处理程序
  echo "<b>异常是:</b> " , $exception->getMessage();
  }
  set_exception_handler('myException');
  throw new Exception('正在处理未被捕获的异常');  //抛出异常信息
?>
```

运行后结果如图 9-13 所示。上面的代码不存在 catch 代码块，而是触发顶层的异常处理程序。用户应该使用此函数来捕获所有未被捕获的异常。

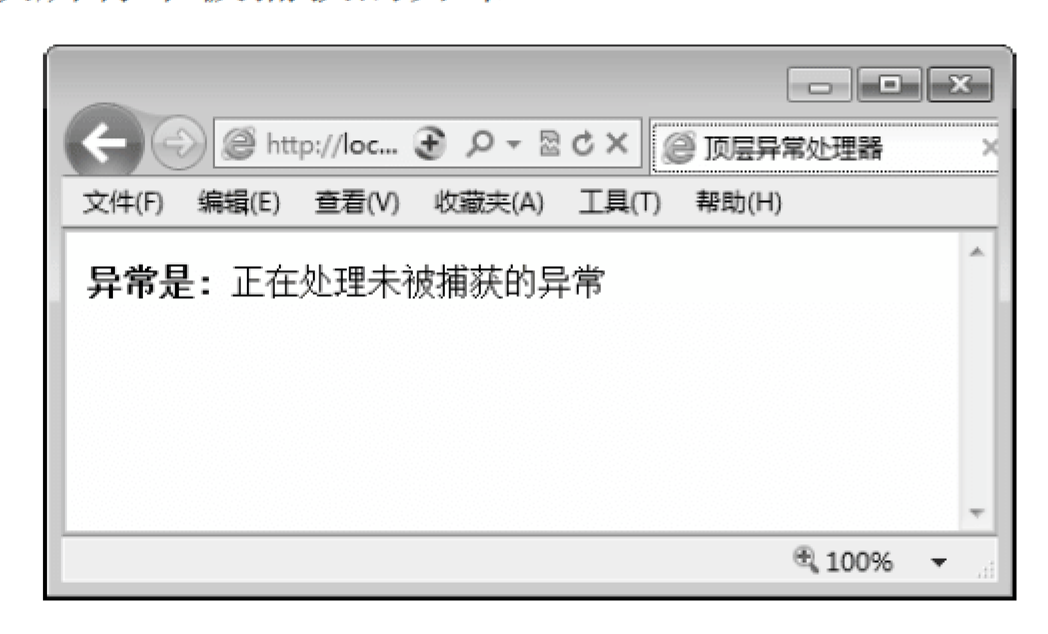


图 9-13 顶层异常处理器

9.4 实战演练——处理异常或错误

错误处理也叫意外处理。通过使用 try…throw…catch 结构和一个内置函数 Exception()来“抛出”和“处理”错误或异常。

下面通过打开文件的实例介绍意外的处理方法和技巧。

【例 9.11】（实例文件：ch10\9.11.php）

```
<?php
 $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];  //定义变量
 @$fp = fopen("$DOCUMENT_ROOT/book.txt",'rb');        //打开指定文件 book.txt
//判断文件是否存在，不存在则抛出异常
 try{
    if (!$fp){
```

```
        throw new Exception("文件路径有误或找不到文件。");
      }
   }catch(Exception $exception){
      echo $exception->getMessage();
      echo "在文件". $exception->getFile()."的".$exception->getLine()."行。<br
/>";
   }
   @fclose($fp);          //屏蔽错误信息
  ?>
```

运行结果如图 9-14 所示。

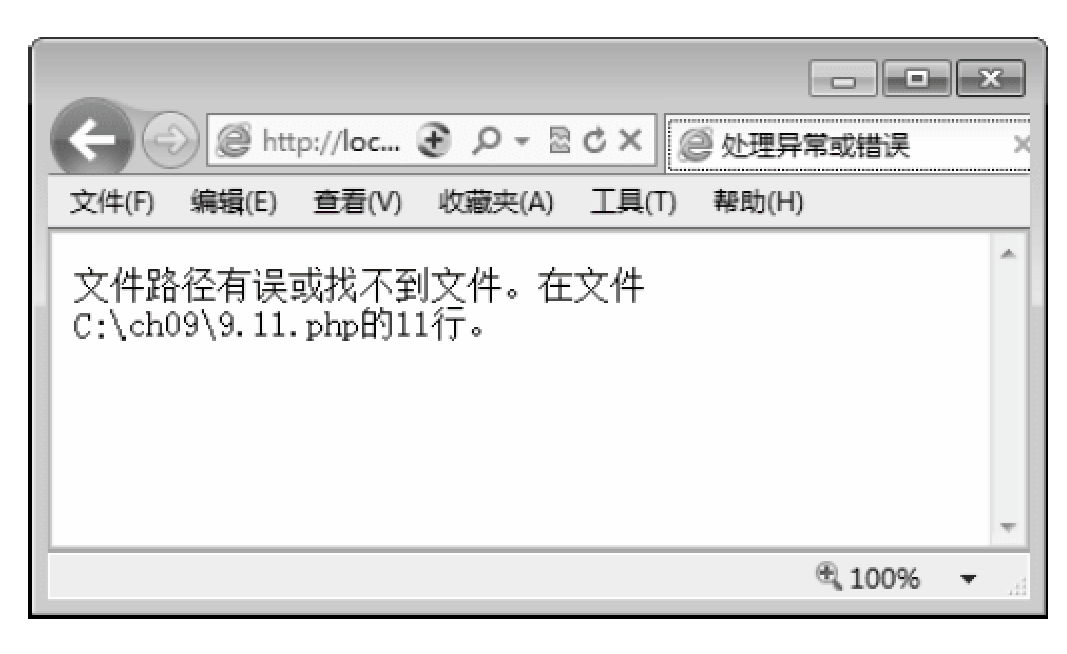


图 9-14　程序运行结果

【案例分析】：

（1）fopen()函数打开$DOCUMENT_ROOT/book.txt 文件进行读取，但是由于 book.txt 文件不存在，则$fp 为 false。

（2）try 区块判断$fp 为 false 时，抛出一个异常。此异常直接通过 new 关键字生成 Exception()类的实例。异常信息是传入参数定义的“文件路径有误或找不到文件”。

（3）catch 区块通过处理传入的 Exception()类实例，显示出错误信息、错误文件、错误发生行。这些是通过直接调用 Exception()类实例$exception 的内置类方法获得。错误信息由 getMessage()生成，错误文件由 getFile()生成，错误发生行由 getLine()生成。

（4）@fclose()和 @$fp= fopen()中的“@”表示屏蔽此命令的执行中产生的错误信息。

9.5　高手私房菜

技巧 1：处理异常有什么规则？

在处理异常时，有以下规则需要用户牢牢掌握。

（1）需要进行异常处理的代码应该放入 try 代码块内，以便捕获潜在的异常。

（2）每个 try 或 throw 代码块必须至少拥有一个对应的 catch 代码块。

（3）使用多个 catch 代码块可以捕获不同种类的异常。

（4）可以在 try 代码块内的 catch 代码块中再次抛出（re-thrown）异常。

技巧 2：如何隐藏错误信息？

PHP 提供了一种隐藏错误的方法，就是在被调用的函数名前加“@”符号，这样会隐藏可能由于这个函数导致的错误信息。

例如以下代码：

```
<?php
  $ab=fopen("123.txt","r");                    //打开指定的文件
  fclose();                                    //关闭指定的文件
?>
```

由于指定的文件不存在，所以运行后会弹出如图 9-15 所示的错误信息。

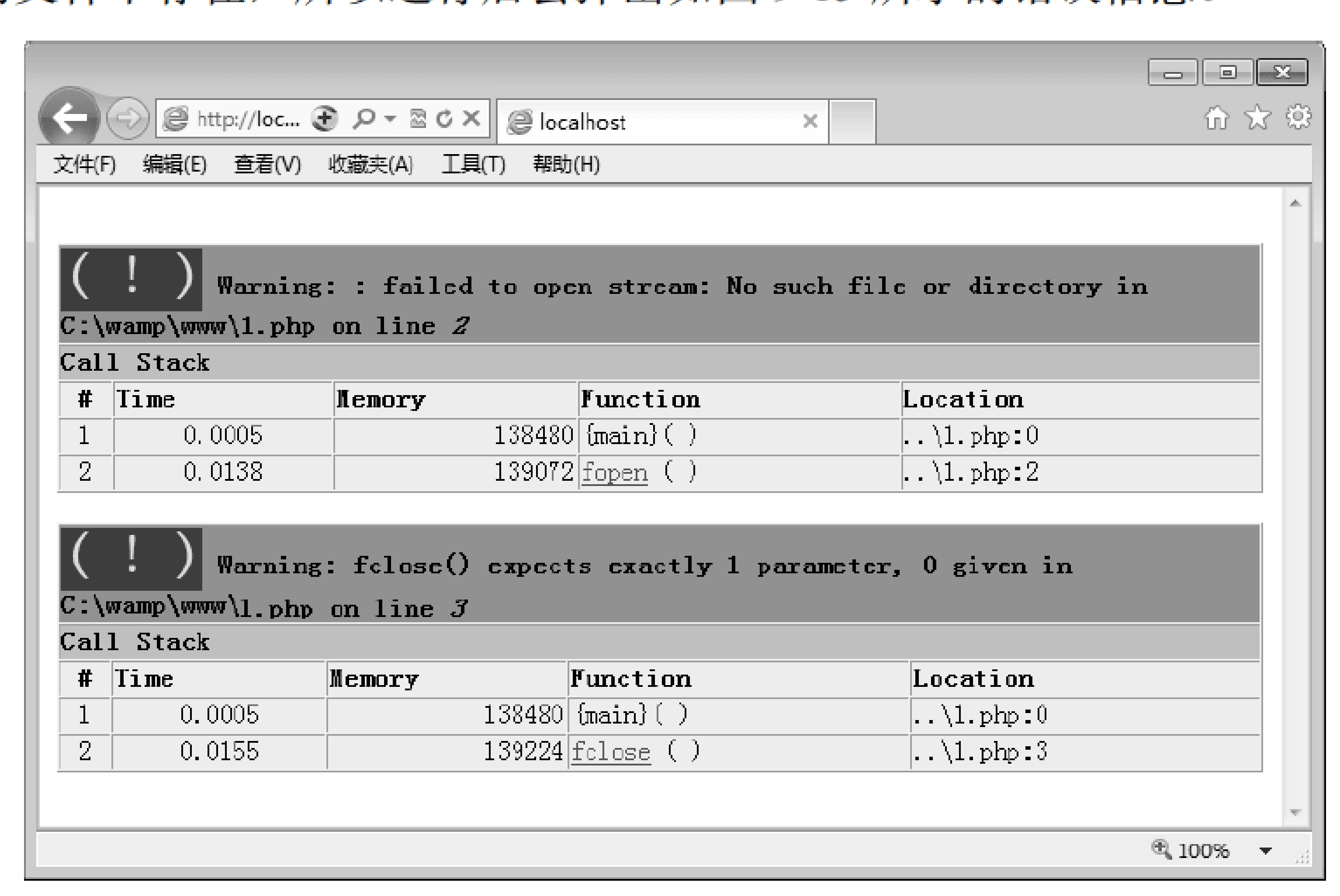


图 9-15　程序运行结果

如果在 fopen()函数和 fclose()函数前加上“@”符号，再次运行程序时，就不会出现上述的错误信息。这种隐藏信息的方法对于查找错误的位置是很有帮助的。

技巧 3：PHP 7 在异常处理方面有什么新变化？

PHP 7 改变了大多数错误的报告方式。不同于 PHP 5 的传统错误报告机制，现在大多数错误被作为 Error 异常抛出。这种 Error 异常可以像普通异常一样被 try/catch 块所捕获。如果没有匹配的 try/catch 块，则按照传统方式处理：被报告为一个致命错误（Fatal Error）。

9.6　经典习题

（1）制作一个包含错误处理的例子。

（2）制作一个包含异常处理的例子。

（3）制作一个处理数据库连接失败的例子。

（4）制作一个处理环境配置错误的例子。

第 10 章　PHP 与 Web 页面的交互

PHP 是一种专门用于 Web 开发的服务器端脚本语言。从这个描述可以知道，PHP 要打交道的对象主要有服务器（server）和基于 Web 的 HTML（超文本标识语言）。使用 PHP 处理 Web 应用时，需要把 PHP 代码嵌入到 HTML 文件中。每次当这个 HTML 网页被访问的时候，其中嵌入的 PHP 代码就会被执行，并且返回给请求浏览器以生成好的 HTML。换句话说，在上述过程中，PHP 就是用来被执行且生成 HTML 的。本章主要讲述 PHP 与 Web 页面的交互操作技术。

本章学习目标

- 了解使用动态内容
- 掌握表单与 PHP 的联系
- 掌握表单设计的方法
- 掌握传递数据的方法
- 掌握获取表单数据的方法
- 掌握对 URL 传递的参数进行编码的方法

10.1　使用动态内容

为什么要使用动态内容呢？因为动态内容可以给网站使用者不同的和实时变化的内容，极大地提高网站的可用性。如果 Web 应用都只是使用静态内容，则 Web 编程完全不用引入 PHP、JSP 和 ASP 等服务器端脚本语言。通俗地说，使用 PHP 语言的主要原因之一，就是使用动态内容。

下面介绍使用动态内容的案例。此例中，在先不涉及变量和数据类型的情况下，将使用 PHP 中的一个内置函数来获得动态内容。此动态内容就是使用 date()函数获得 Web 服务器的时间。

【例 10.1】（实例文件：ch10\10.1.php）

```
<HTML>
<HEAD>
<h2>
PHP Tells time. - PHP 告诉我们时间。
</h2>
</HEAD>
<BODY>
  <?php date_default_timezone_set("PRC");
   echo "现在的时间为：";
 echo date("H:i:s Y m d");
  ?>
</BODY>
</HTML>
```

运行结果如图 10-1 所示。过一段时间再次运行上述 PHP 页面，即可看到显示的内容发生了动态的变化，如图 10-2 所示。

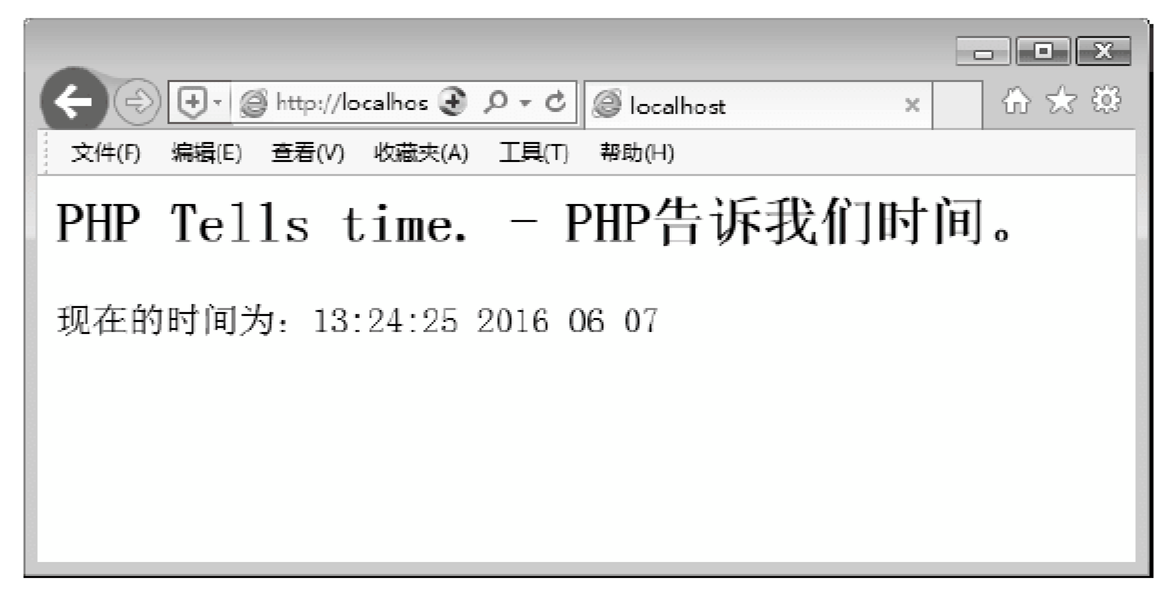


图 10-1　程序运行结果

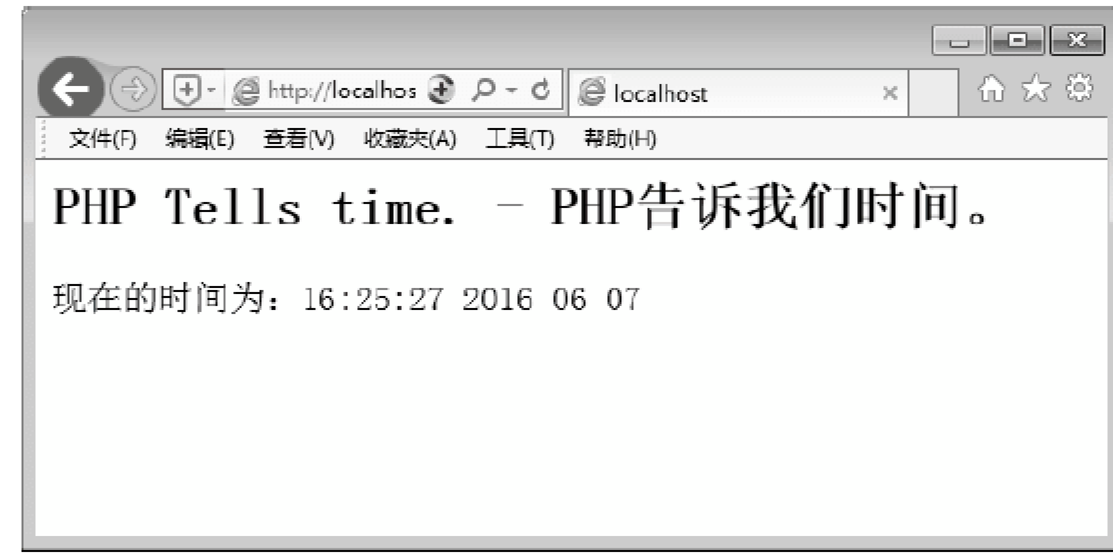


图 10-2　时间发生变化

【案例分析】：

（1）“PHP Tells time. - PHP 告诉我们时间。”是 HTML 中的"<HEAD><h2>PHP Tells time. - PHP 告诉我们时间。</h2></HEAD>"所生成的。后面的“现在的时间为： 13:24:26 2016 06 07”是由"<?php echo "现在的时间为： "; echo date("H:i:s Y m d"); ?>"生成的。

（2）由于“现在的时间为：13:24:26 2016 06 07”是由 date( )函数动态生成并且实时更新的。如果再次打开或刷新此文件，PHP 代码将被再次执行，所输出的时间也会发生改变。

（3）此实例中通过 date()函数处理系统时间，得到动态内容。时间处理是 PHP 中一项重要的功能。

10.2　表单与 PHP

不管是一般的企业网站还是复杂的网络应用，都离不开数据的添加。通过 PHP 服务器端脚本语言，程序可以处理那些通过浏览器对 Web 应用进行数据调用或添加的请求。

回忆一下平常使用的网站数据输入功能，不管是 Web 邮箱还是 QQ 留言，都经常要填一些表格，再由这些表格把数据发送出去。而完成这个工作的部件就是“表单（form）”。

虽然表单（form）是 html 语言的东西，但是 PHP 与 form 变量的衔接是无缝的。PHP 关心的是怎么获得和使用 form 中的数据。由于 PHP 功能强大，可以很轻松地对它们进行处理。

处理表单数据的基本过程是：数据从 Web 表单（form）发送到 PHP 代码，经过处理再生成 html 输出。它的处理原理是：当 PHP 处理一个页面的时候，会检查 URL、表单数据、上传文件、可用 cookie、Web 服务器和环境变量，如果有可用信息，就可以通过 PHP 访问自动全局变量数组$_GET、$_POST、$_FILES、$_COOKIE、$_SERVER 和$_ENV 得到。

10.3　表单设计

表单是一个比较特殊的组件，在 html 中有着比较特殊的功能与结构。下面了解一下表单的基本元素。

10.3.1 表单基本结构

表单的基本结构是由<form></form>标识包裹的区域，例如：

```
<HTML>
<HEAD></HEAD>
<BODY>
<form action=" " method=" " enctype=" " >
    ……
</form>
</BODY>
</HTML>
```

其中，<form >标识内必须包含属性。action 指定数据所要发送的对象文件，method 指定数据传输的方式。如果在上传文件等操作，还要定义 enctype 属性以指定数据类型。

10.3.2 文本框

文本框是 form 输入框中最为常见的。下面通过例子讲述文本框的使用方法。

01 在网站根目录下创建 phpform 文件夹，然后在其下创建文件 formdemo.html，文件代码如下。

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<form action="formdemohandler.php" method="post">
  <h3>输入一个信息（比如名称）：</h3>
  <input type="text" name="name" size="10" />
</form>
</BODY>
</HTML>
```

02 在 phpform 文件夹下创建文件 formdemohandler.php，文件代码如下。

```
<?php
  $name = $_POST['name'];
  echo $name;
?>
```

运行 formdemo.html，结果如图 10-3 所示。



图 10-3 程序运行结果

【案例分析】：

（1）<input type="text" name="name" size="10" />语句定义了 form 的文本框。定义一个输入框为文本框的必要因素为：

```
<input type="text" …… />
```

其他的属性则如实例中一样，可以定义文本框的 name 属性，以确认此文本框的唯一性，定义 size 属性以确认此文本框的长度。

（2）在 formdemohandler.php 文件中，则使文本框的 name 值为 'name'。

10.3.3 选项框

复选框可用于选择一项或者多项。通过修改 formdemo 的例子加以说明，具体操作如下。

01 在 phpform 文件夹下修改文件 formdemo.html 为如下代码。

```
<HTML>
<HEAD></HEAD>
<BODY>
<form action="formdemohandler.php" method="post">
  <h3>输入一个信息（比如名称）：</h3>
  <input type="text" name="name" size="10" />
  <h3>确认此项（可复选）：</h3>
  <input type="checkbox" name="achecked" checked="checked" value="1" />
  选择此项传递的 A 项的 value 值。
  <input type="checkbox" name="bchecked"  value="2" />
  选择此项传递的 B 项的 value 值。
  <input type="checkbox" name="cchecked"  value="3" />
  选择此项传递的 C 项的 value 值。
</form>
</BODY>
</HTML>
```

02 在 phpform 文件夹下修改文件 formdemohandler.php，其代码如下。

```
<?php
  $name = $_POST['name'];
  if(isset($_POST['achecked'])){
    $achecked = $_POST['achecked'];
  }
  if(isset($_POST['bchecked'])){
  $bchecked = $_POST['bchecked'];
  }
  if(isset($_POST['cchecked'])){
  $cchecked = $_POST['cchecked'];
  }
  $aradio = $_POST['aradio'];
```

```
    $aselect = $_POST['aselect'];

    echo $name."<br />";

    if(isset($achecked) and $achecked == 1){
       echo "选项 A 的 value 值已经被正确传递。<br />";
    }else{
       echo "选项 A 没有被选择，其 value 值没有被传递。<br />";
    }
    if(isset($bchecked) and $bchecked == 2){
       echo "选项 B 的 value 值已经被正确传递。<br />";
    }else{
       echo "选项 B 没有被选择，其 value 值没有被传递。<br />";
    }
    if(isset($cchecked) and $cchecked == 3){
       echo "选项 C 的 value 值已经被正确传递。<br />";
    }else{
       echo "选项 C 没有被选择，其 value 值没有被传递。<br />";
    }
  ?>
```

03 运行 formdemo.html，结果如图 10-4 所示。

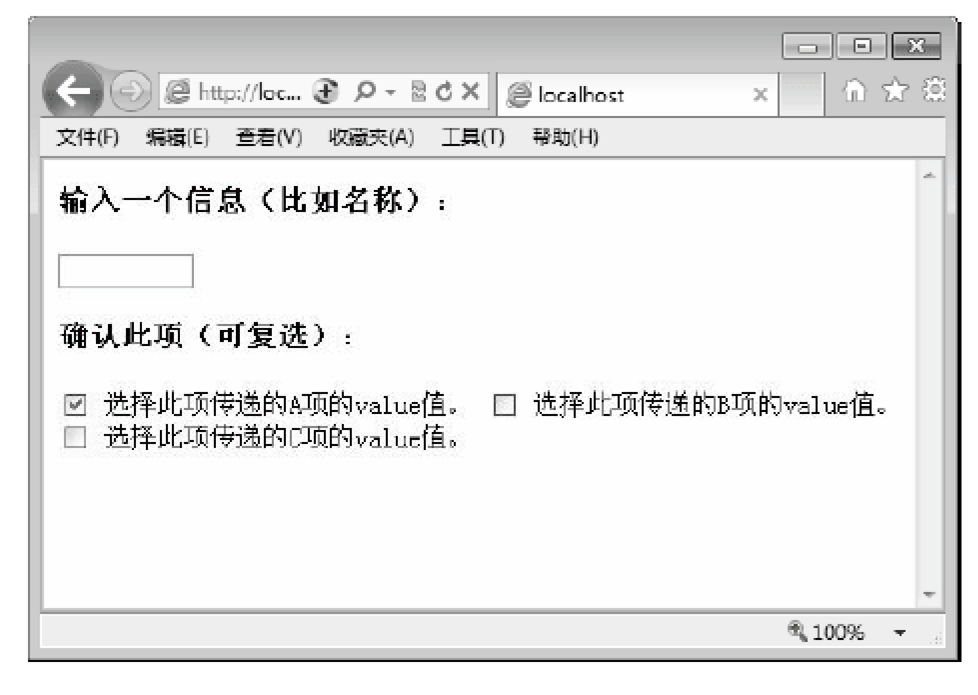


图 10-4 程序运行结果

【案例分析】：

（1）<input type="checkbox" name="achecked" checked="checked" value="1" />语句定义了复选框。定义一个 input 标识为复选框时需指定类型为 checkbox：

```
  <input type="checkbox" …… />
```

定义为复选框之后，还需要定义复选框的 name 属性，以确定在服务器端程序的唯一性；定义 value 属性，以确定此单选项所要传递的值；定义 Checked 属性，以确定复选框的默认状态，若为 checked 则默认为选择，如果不定义此项，默认为不选择。

（2）在 formdemohandler.php 文件中，则使选项的 name 值为'achecked'、name 值为'bchecked'、name 值为'cchecked' 并且根据 value 值作出判断。

10.3.4 单选按钮

下面通过案例来介绍如何使用单选按钮，仍然通过修改 formdemo 的例子加以说明，具体步骤如下。

01 在 phpform 文件夹下修改文件 formdemo.html，代码如下。

```
<HTML>
<HEAD></HEAD>
<BODY>
<form action="formdemohandler.php" method="post">
……
   <h3>单选一项：</h3>
   <input type="radio"  name="aradio" value="a1" />蓝天
   <input type="radio"  name="aradio" value="a2" checked="checked" />白云
   <input type="radio"  name="aradio" value="a3" />大海
</form>
</BODY>
</HTML>
```

02 在 phpform 文件夹下修改文件 formdemohandler.php，代码如下。

```
<?php
……
  $aradio = $_POST['aradio'];

  echo $name;
……
  if(isset($achecked) and $cchecked == 3){
     echo "选项 C 的 value 值已经被正确传递。<br />";
  }else{
     echo "选项 C 没有被选择，其 value 值没有被传递。<br />";
  }
  if($aradio == 'a1'){
     echo "蓝天";
  }else if($aradio == 'a2'){
     echo "白云";
  }else{
     echo "大海";
  }
?>
```

03 运行 formdemo.html，结果如图 10-5 所示。

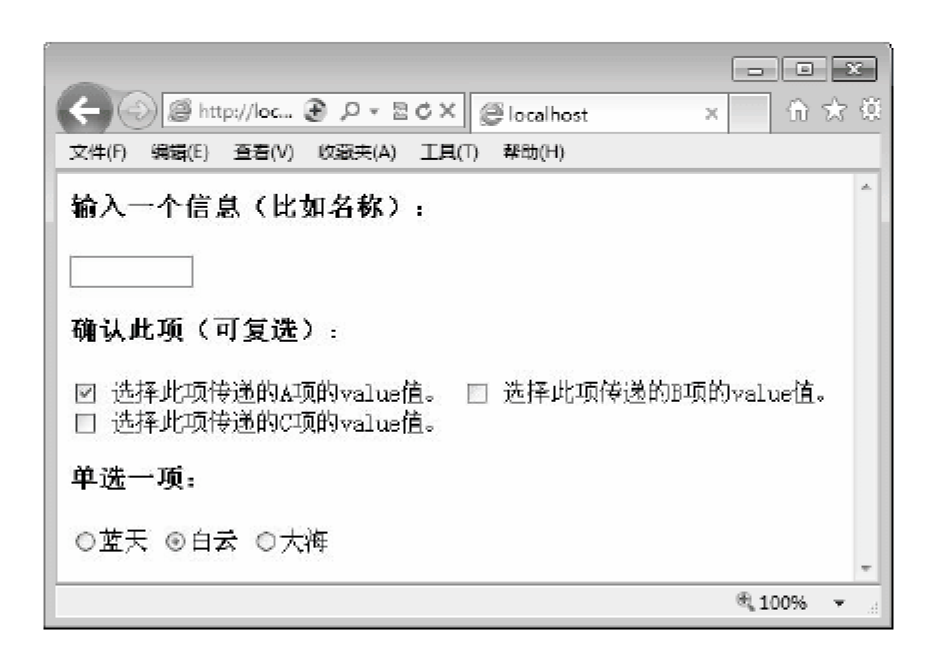


图 10-5　程序运行结果

【案例分析】：

（1）<input type="radio"　name="aradio" value="a1" />语句定义了一个单选按钮。后面的<input type="radio"　name="aradio" value="a2" checked="checked" />和<input type="radio"　name="aradio" value="a3" />定义了另外两个单选按钮。

定义一个 input 标识为单选按钮时需指定类型为 radio：

```
<input type="radio" …… />
```

定义为单选按钮之后，还需要定义单选按钮的 name 属性，以确定在服务器端程序的唯一性；定义 value 属性，以确定此单选按钮所要传递的值；定义 checked 属性，以确定单选按钮的默认状态，若为 checked 则默认为选择，如果不定义此项，默认为不选择。

（2）在 formdemohandler.php 文件中，则使单选按钮的 name 值为"aradio"，然后 if 语句通过对 aradio 传递的不同的值作出判断，打印不同的值。

10.3.5　下拉列表

下面通过实例来介绍下拉列表的使用方法和技巧，仍然通过修改 formdemo 的例子加以说明，具体步骤如下。

01 在 phpform 文件夹下修改文件 formdemo.html，添加代码如下。

```
<HTML>
<HEAD></HEAD>
<BODY>
<form action="formdemohandler.php" method="post">
 ……
   <h3>在下拉菜单中选择一项：</h3>
 <select name="aselect" size="1">
     <option value="hainan">海南</option>
     <option value="qingdao" selected>青岛</option>
     <option value="beijing">北京</option>
     <option value="xizang">西藏</option>
 </select>
</form>
</BODY>
```

```
</HTML>
```

02 在 phpform 文件夹下修改文件 formdemohandler.php，代码如下。

```
<?php
……
  $aselect = $_POST['aselect'];

  echo $name."<br />";
  ……
  }else{
     echo "大海";
  }
  if($aselect == 'hainan'){
     echo "海南";
  }else if($aselect == 'qingdao'){
     echo "青岛";
  }else if($aselect == 'beijing'){
     echo "北京";
  }else{
     echo "西藏";
  }

?>
```

03 运行 formdemo.html，结果如图 10-6 所示。

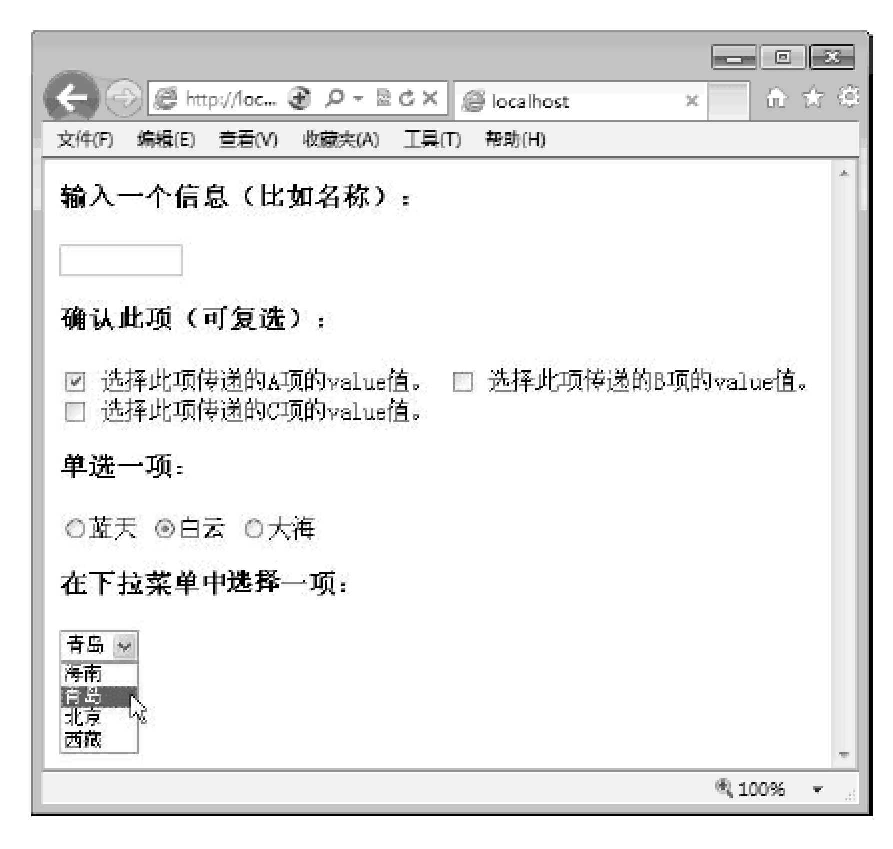


图 10-6　程序运行结果

【案例分析】：

（1）下拉列表是通过<select></select>标识表示的。而下拉列表当中的选项是通过包含在其中的<option></option>标识表示的。<select>标识中 name 定义下拉列表的 name 属性，以确认它的唯一性。<option>标识中 value 定义需要传递的值。

（2）在 formdemohandler.php 文件中，则使选项的 name 值为"aselect"。然后 if 语句通过对 aselect 传递的不同的值做出判断，打印不同的值。

10.3.6　重置按钮

重置按钮用来重置所有的表单输入的数据。对于重置按钮的使用，仍然通过修改 formdemo 的例子加以说明，具体步骤如下。

01 在文件夹下修改文件 formdemo.html，代码如下。

```
<HTML>
<HEAD></HEAD>
<BODY>
<form action="formdemohandler.php" method="post">
……
 <h3>点击此按钮重置所有信息：</h3>
   <input type="RESET" value="重置">
</form>
</BODY>
</HTML>
```

02 运行 formdemo.html，结果如图 10-7 所示。

03 点击“重置”按钮，页面中所有输入数据被重置为默认值，如图 10-8 所示。

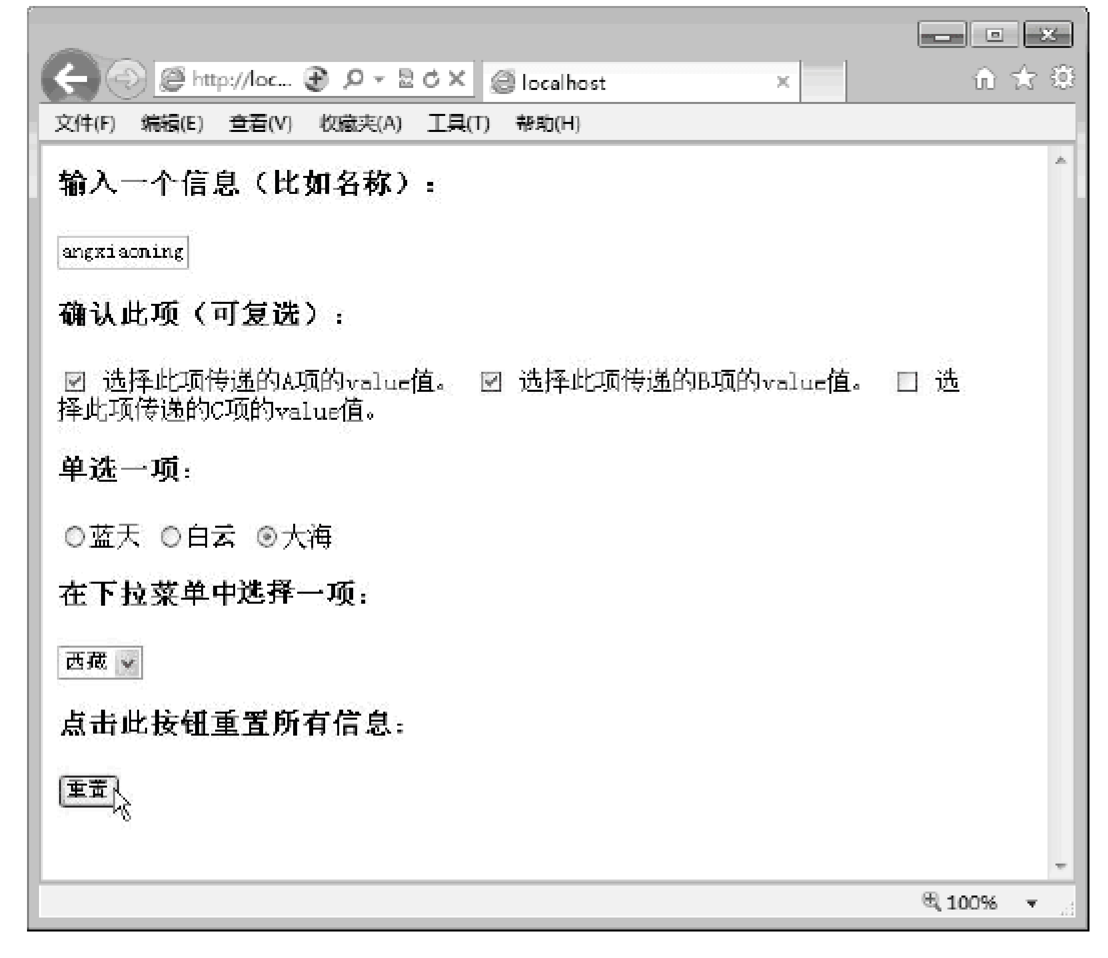


图 10-7　程序运行结果

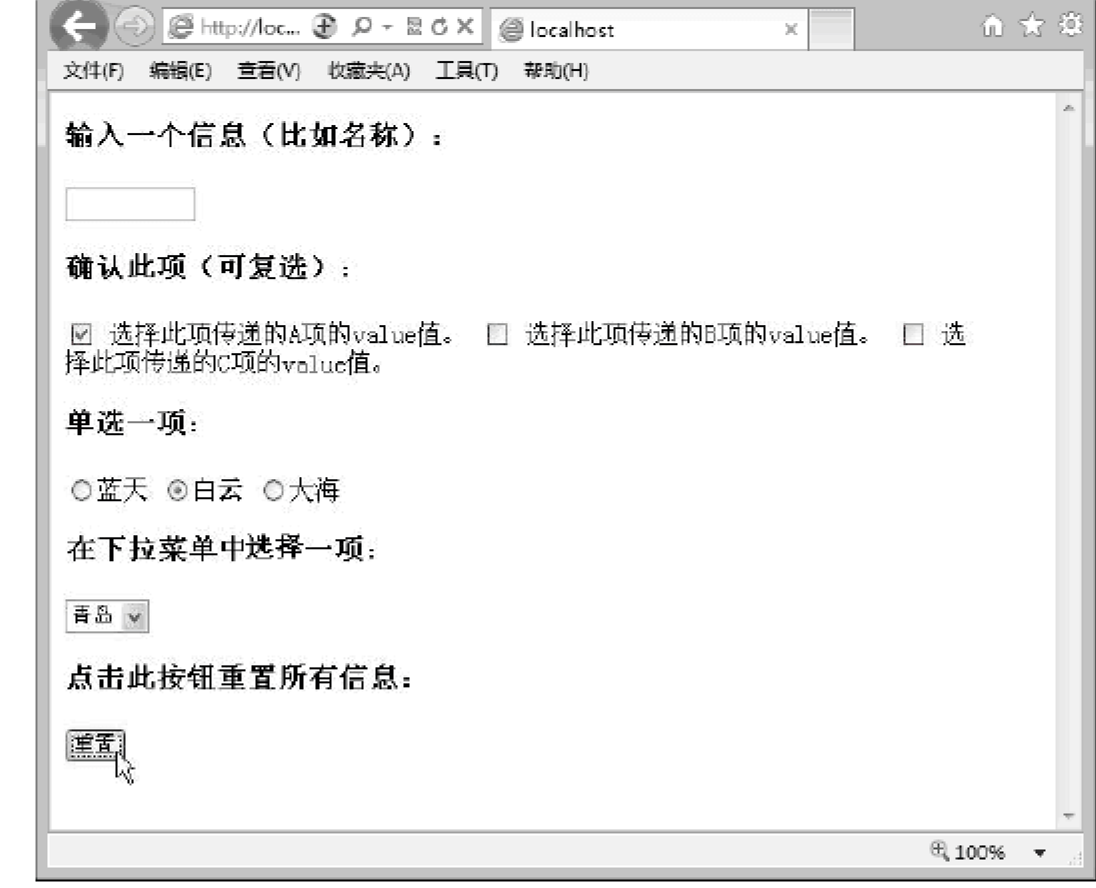


图 10-8　重置为默认值

由<input type="RESET" value="重置">语句可见，重置按钮是<input />标识的一种。定义一个 input 标识为单选项的必要因素为：

```
<input type="reset" …… />
```

Value 属性是按钮所显示的字符。

10.3.7　提交按钮

到现在为止，上面程序中 form 中的所有元素都已经设置完成，并且在相应的 PHP 文件中做了处理。这个时候，要想把 HTML 页面中所有的数据发送出去给相应 PHP 文件进行处理，就需要使

用 submit 按钮，也就是“提交”按钮。下面添加“提交”按钮，并且提交数据。具体步骤如下。

01 在 phpform 文件夹下修改文件 formdemo.html，最终代码如下。

```
<HTML>
<HEAD></HEAD>
<BODY>
<form action="formdemohandler.php" method="post">
  <h3>输入一个信息（比如名称）：</h3>
  <input type="text" name="name" size="10" />
  <h3>确认此项（可复选）：</h3>
  <input type="checkbox" name="achecked" checked="checked" value="1" />
  选择此项传递的 A 项的 value 值。
  <input type="checkbox" name="bchecked"  value="2" />
  选择此项传递的 B 项的 value 值。
  <input type="checkbox" name="cchecked"  value="3" />
  选择此项传递的 C 项的 value 值。
   <h3>单选一项：</h3>
   <input type="radio"  name="aradio" value="a1" />蓝天
   <input type="radio"  name="aradio" value="a2" checked="checked" />白云
   <input type="radio"  name="aradio" value="a3" />大海
   <h3>在下拉菜单中选择一项：</h3>
 <select name="aselect" size="1">
     <option value="hainan">海南</option>
     <option value="qingdao" selected>青岛</option>
     <option value="beijing">北京</option>
     <option value="xizang">西藏</option>
 </select>
 <h3>点击此按钮重置所有信息：</h3>
   <input type="RESET" value="重置" />
   <h3>点击此按钮提交所有信息到 formdemohandler.php 文件：</h3>
   <input type="submit" value="提交" />
</form>
</BODY>
</HTML>
```

02 在 phpform 文件夹下修改文件 formdemohandler.php，其最终代码如下。

```
<?php
  $name = $_POST['name'];
  if(isset($_POST['achecked'])){
    $achecked = $_POST['achecked'];
  }
  if(isset($_POST['bchecked'])){
  $bchecked = $_POST['bchecked'];
  }
  if(isset($_POST['cchecked'])){
```

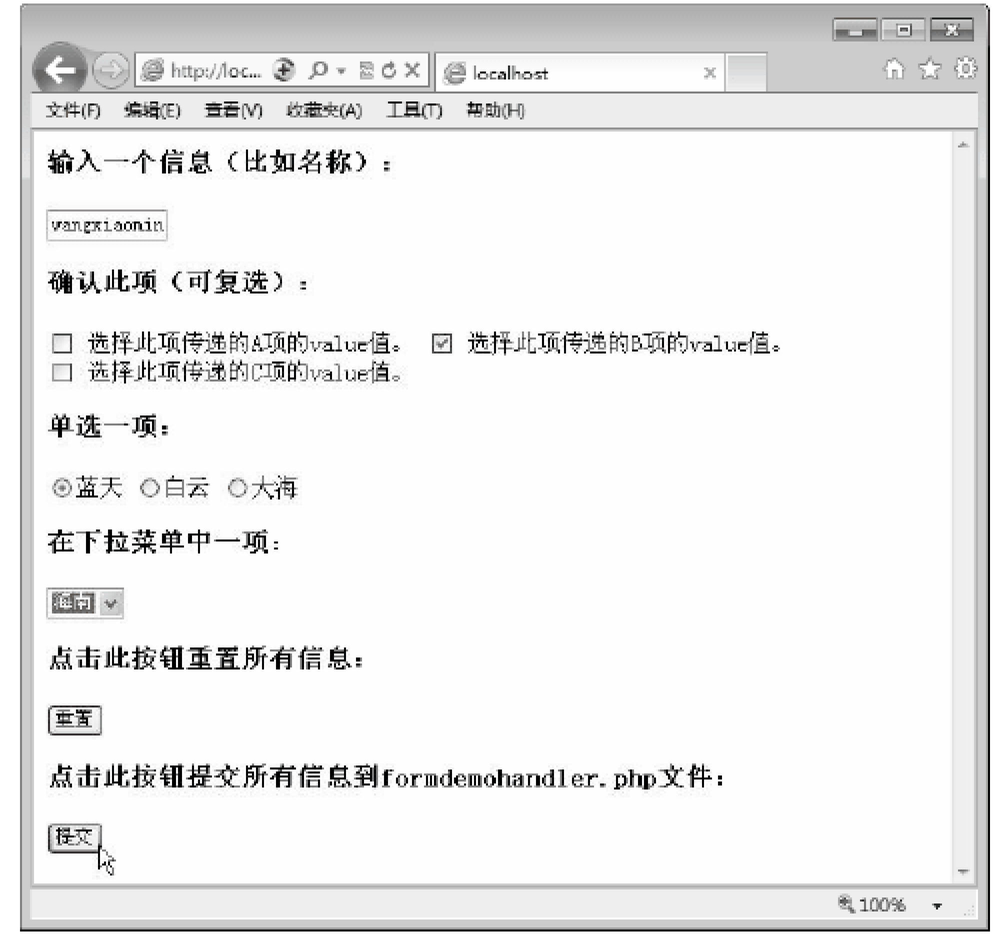
```
    $cchecked = $_POST['cchecked'];
    }
    $aradio = $_POST['aradio'];
    $aselect = $_POST['aselect'];
    echo $name."<br />";
    if(isset($achecked) and $achecked == 1){
       echo "选项 A 的 value 值已经被正确传递。<br />";
    }else{
       echo "选项 A 没有被选择，其 value 值没有被传递。<br />";
    }
    if(isset($bchecked) and $bchecked == 2){
       echo "选项 B 的 value 值已经被正确传递。<br />";
    }else{
       echo "选项 B 没有被选择，其 value 值没有被传递。<br />";
    }
    if(isset($cchecked) and $cchecked == 3){
       echo "选项 C 的 value 值已经被正确传递。<br />";
    }else{
       echo "选项 C 没有被选择，其 value 值没有被传递。<br />";
    }

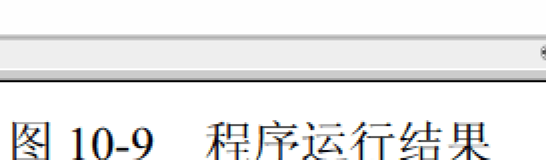
    if($aradio == 'a1'){
       echo "蓝天<br />";
    }else if($aradio == 'a2'){
       echo "白云<br />";
    }else{
       echo "大海<br />";
    }

    if($aselect == 'hainan'){
       echo "海南<br />";
    }else if($aselect == 'qingdao'){
       echo "青岛<br />";
    }else if($aselect == 'beijing'){
       echo "北京<br />";
    }else{
       echo "上海";
    }
?>
```

03 运行 formdemo.html，结果如图 10-9 所示。

04 单击“提交”按钮，页面跳转到 formdemohandler.php，输出结果如图 10-10 所示。





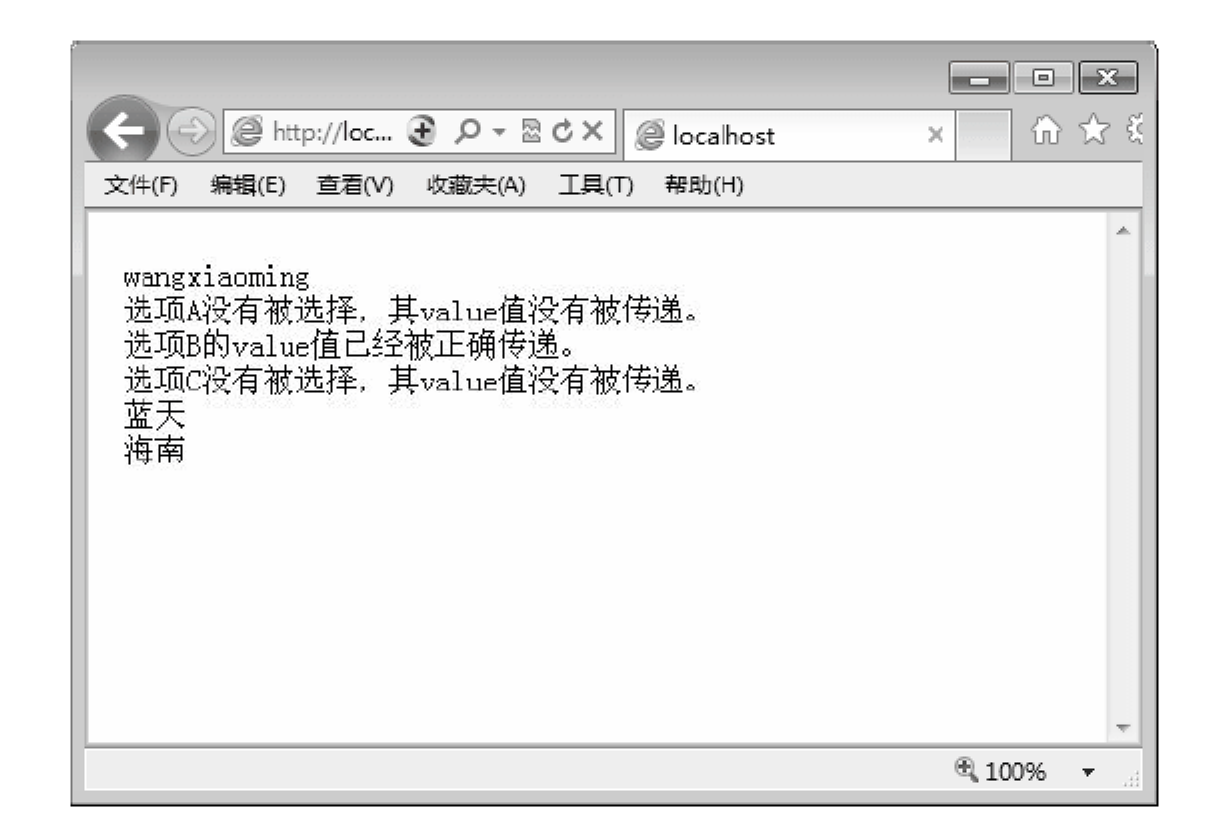


图 10-9　程序运行结果　　　　图 10-10　提交数据

10.4　传递数据的两种方法

数据传递的常用方法有 POST 和 GET 两种，下面来介绍这两种方法的使用技巧。

10.4.1　用 POST 方式传递数据

表单（Form）传递数据是通过 POST 和 GET 两种方式进行的。在定义表单属性的时候，要在 method 属性上定义使用哪种数据传递方式。

<form action="URI" method="post">定义了表单在把数据传递给目标文件的时候，使用的是 POST 方式。<form action="URI" method="get">则定义了表单在把数据传递给目标文件的时候，使用的是 GET 方式。

POST 是比较常见的表单提交方式。通过 POST 方式提交的变量，不受特定的变量大小的限制，并且被传递的变量不会在浏览器地址栏里以 URL 的方式显示出来。

10.4.2　用 GET 方式传递数据

GET 方式比较有特点。通过 GET 方式提交的变量有大小限制，不能超过 100 个字符。它的变量名和与之相对应的变量值都会以 URL 的方式显示在浏览器地址栏里。所以，若传递大而敏感的数据，一般不使用此方式。

使用 GET 方式传递数据，通常使用 URL 连接来进行的。

下面对此操作进行讲解，具体步骤如下。

01 在网站根目录下建立 getparam.php 文件，输入以下代码并保存。

```
<?php
if(!$_GET['u'])
{
  echo '参数还没有输入。';
}else{
    $user=$_GET['u'];
```

```
    switch ($user){
     case 1:
         echo "用户是王小明";
          break;
      case 2:
         echo "用户是李丽丽";
          break;
    }
}
?>
```

02 在浏览器地址栏中输入“http://localhost/getparam.php?u”，并按回车键确认，运行结果如图 10-11 所示。

03 在浏览器地址栏中输入“http://localhost/getparam.php?u=1”，并按回车键确认，运行结果如图 10-12 所示。



图 10-11　程序运行结果



图 10-12　程序运行结果

04 在浏览器地址栏中输入“http://localhost/getparam.php?u=2”，并按回车键确认，运行结果如图 10-13 所示。



图 10-13　程序运行结果

【案例分析】：

（1）在 URL 中 GET 方式通过“？”号后面的数组元素的键名（这里是“u”）来获得元素值。

（2）对元素赋值，使用“=”号。

（3）switch 条件语句做出判断并返回结果。

10.5　PHP 获取表单传递数据的方法

如果表单使用 POST 方式传递数据，则 PHP 要使用全局变量数组$_POST[]来读取所传递的数据。

表单中元素传递数据给$_POST[]全局变量数组，其数据以关联数组中的数组元素形式存在。以表单元素的名称属性为键名，以表单元素的输入数据或传递的数据为键值。

比如前面 formdemohandler.php 文件中$name = $_POST['name'];语句就是读取名为 name 的文本框中的数据。此数据以 name 为键名，以文本框输入的数据为键值。

再如$achecked = $_POST['achecked']语句，读取名为 achecked 的复选框传递的数据。此数据以 achecked 为键名，以复选框传递的数据为键值。

如果表单使用 GET 方式传递数据，则 PHP 要使用全局变量数组$_GET[]来读取所传递的数据。与$_POST[]相同，表单中元素传递数据给$_GET[]全局变量数组，其数据以关联数组中的数组元素形式存在。以表单元素的名称属性为键名，以表单元素的输入数据或是传递的数据为键值。

10.6　PHP 对 URL 传递的参数进行编码

PHP 对 URL 中传递的参数进行编码，一可以实现对所传递数据的加密，二可以对无法通过浏览器传递的字符进行传递。要实现此操作一般使用 urlencode()和 rawurlencode()函数。而对此过程的反向操作就是使用 urldecode()和 rawurldecode()函数。

下面对此操作进行讲解，具体步骤如下。

01 在网站根目录下建立 urlencode.php 文件，输入以下代码并保存。

```
<?php
$user = '王小明 刘晓莉';                                   //定义变量
$link1 = "index.php?userid=".urlencode($user)."<br />";     //对字符串进行编码
$link2 = "index.php?userid=".rawurlencode($user)."<br />";  //对字符串进行编码
echo $link1.$link2;
//对编码的反向操作
echo urldecode($link1);
echo urldecode($link2);
echo rawurldecode($link2);
?>
```

02 在浏览器地址栏中输入“http://localhost/urlencode.php”，并按回车键确认，运行结果如图 10-14 所示。

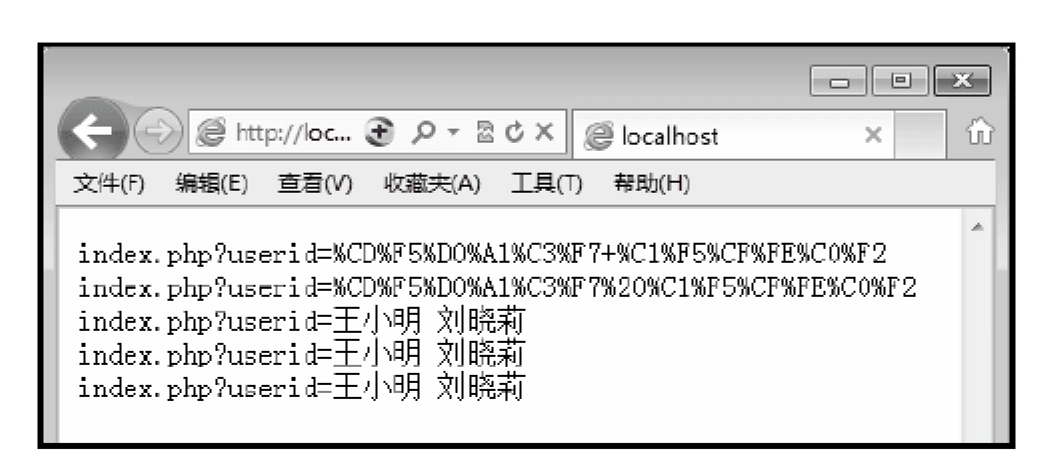


图 10-14　程序运行结果

【案例分析】:

（1）在$link1 变量的赋值中，使用 urlencode()函数对一个中文字符串$user 进行编码。

（2）在$link2 变量的赋值中，使用 rawurlencode()函数对一个中文字符串$user 进行编码。

（3）这两种编码方式的区别在于对空格的处理，urlencode()函数将空格编码为“+”号，而 rawurlencode()函数将空格编码为“%20”加以表述。

（4）urldecode()函数实现对编码的反向操作。

10.7 实战演练——PHP 与 Web 表单的综合应用

下面进行处理表单数据的讲解。此实例将假设一名网络浏览者在某酒店网站上登记房间。具体步骤如下。

01 在网站根目录下建立一个 html 文件 form.html，输入以下代码并保存。

```
<HTML>
<HEAD><h2>GoodHome online booking form. - GoodHome 在线订房表。</h2></HEAD>
<BODY>
<form action="formhandler.php" method="post">
<table>
<tr bgcolor="#3399FF">
   <td>客人姓名:</td>
   <td><input type="text" name="customername" size="10" /></td>
</tr>
<tr bgcolor="#CCCCCC" >
   <td>到达时间:</td>
   <td><input type="text" name="arrivaltime" size="3" />天内</td>
</tr>
<tr bgcolor="#3399FF" >
   <td>联系电话:</td>
   <td><input type="text" name="phone" size="15" /></td>
</tr>
<tr bgcolor="#666666" >
   <td align="center"><input type="submit" value="确认订房信息" /></td>
</tr>
</table>
</form>
</BODY>
</HTML>
```

02 在浏览器地址栏中输入“http://localhost/form.html”，并按回车键确认，运行结果如图 10-15 所示。

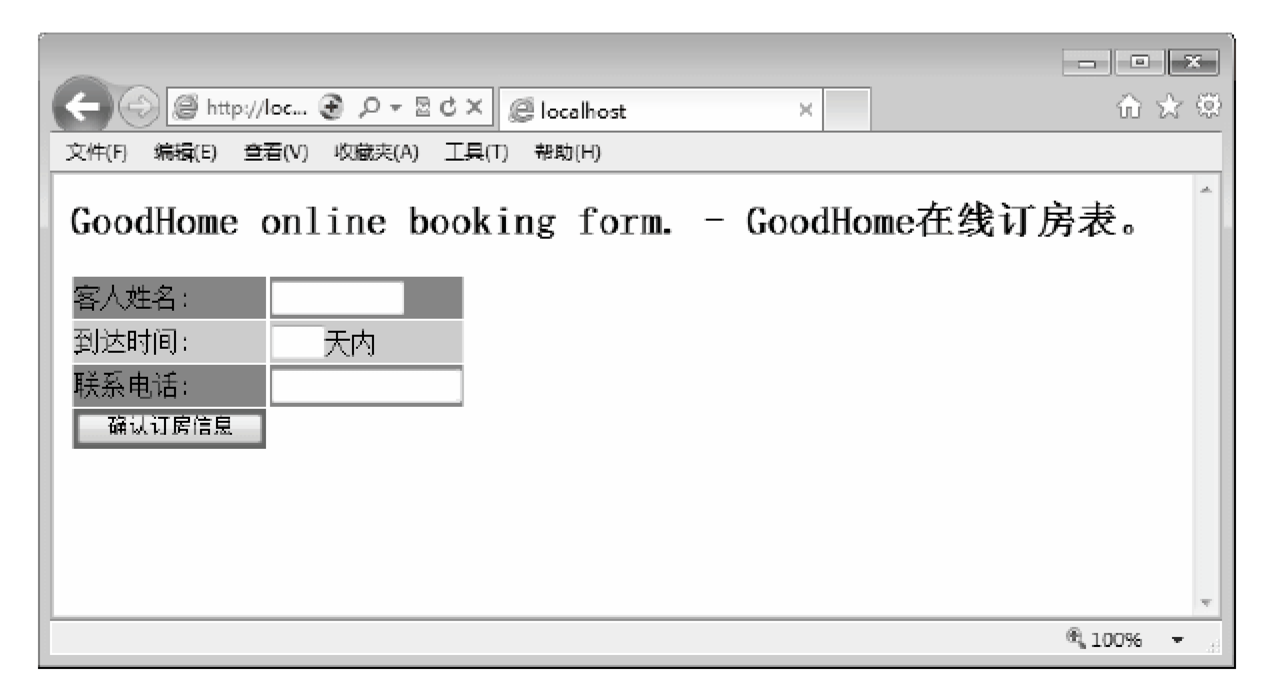


图 10-15　程序运行结果

03 在相同目录下建立一个 PHP 文件 formhandler.php，输入以下代码并保存。

```
<HTML>
<HEAD>
<H2>GoodHome booking info. - GoodHome 订房表确认信息。</H2>
</HEAD>
<BODY>
<?php
  $customername = $_POST['customername'];
  $arrivaltime = $_POST['arrivaltime'];
  $phone = $_POST['phone'];

  echo '<p>订房确认信息:</p>';
  echo '客人 '.$customername.' 您将会在 '.$arrivaltime.' 内到达。 您的联系电话是
'.$phone'。';
?>
</BODY>
</HTML>
```

04 回到浏览器中打开的 form.html 页面。在表单中输入数据，【客人姓名】为“王小明”、【到达时间】为“3”、【联系电话】为“1359XXXX377”，单击【确认订房信息】按钮，浏览器会自动跳转到 formhandler.php 页面，显示结果如图 10-16 所示。



图 10-16　程序运行结果

【案例分析】：

（1）在 form.html 中 form 通过 POST 方法（method）把 3 个<input type="text" …. />中的文本数据发送给 formhandler.php。

（2）在 formhandler.php 中，代码通过读取数组$_POST 中的具体变量$_POST['customername']、$_POST['arrivaltime']和$_POST['phone']，把它赋值给本地变量$customername、$arrivaltime 和$phone。然后，通过 echo 命令使用本地变量，把信息生成 html 后输出给浏览器。

（3）要提到的是"echo '客人 '.$customername.'，您将会在 '.$arrivaltime.' 天内到达。 您的联系电话是 '.$phone.'。';" 中的“.”是字符串连接操作符，它把不同部分的字符串连接在一起。在使用 echo 命令的时候经常会用到它。

10.8　高手私房菜

技巧 1：使用 urlencode()和 rawurlencode()函数需要注意什么？

如果配合 JS 处理页面的信息的话，要注意 urlencode()函数使用后，“+”号与 JS 的冲突。由于 JS 中“+”号是字符串类型的连接操作符。JS 在处理 URL 时就无法识别其中的“+”号。这时可以使用 rawurlencode()函数对其进行处理。

技巧 2：理解 GET 和 POST 的区别和联系。

两者的区别与联系如下。

（1）POST 是向服务器传送数据；GET 是从服务器上获取数据。

（2）POST 是通过 HTTP POST 机制，将表单内各个字段与其内容放置在 HTML HEADER 中一起传送到 ACTION 属性所指的 URL 地址。用户看不到这个过程；GET 是把参数数据队列添加到提交表单的 ACTION 属性所指的 URL 中，值和表单内各个字段一一对应，在 URL 中可以看到。

（3）对于 GET 方式，服务器端用 Request.QueryString 获取变量的值；对于 POST 方式，服务器端用 Request.Form 获取提交的数据。

（4）POST 传送的数据量较大，一般被默认为不受限制。但理论上，IIS4 中最大量为 80KB，IIS5 中为 100KB；GET 传送的数据量较小，不能大于 2KB。

（5）POST 安全性较高；GET 安全性非常低，但是执行效率却比 POST 方法好。

（6）在做数据添加、修改或删除时，建议用 POST 方式；而在做数据查询时，建议用 GET 方式。

（7）机密信息的数据，建议用 POST 数据提交方式。

10.9　经典习题

（1）制作一个包含各种表单元素注册表的例子。

（2）制作两个传递数据方式的例子，并分析它们的区别。

（3）制作一个 PHP 获取表单数据的例子。

（4）制作一个 PHP 对 URL 传递参数进行编码的例子。

第 11 章　PHP 文件与目录操作

前面的章节中，已经实现了用 form 发送数据给 PHP，PHP 再处理数据并输出 html 给浏览器。在这样的流程里，数据会直接被 PHP 代码处理成 html。如果想把数据存储起来，并在需要的时候读取和处理，该怎么办呢？这就是本章需要解决的问题。在 PHP 开发网站的过程中，文件的操作大致分为对普通文件的操作和对数据库文件的操作。本章主要讲述如何对普通文件进行写入和读取、目录的操作、文件的上传等操作。

本章学习目标

- 掌握文件的基本操作方法
- 掌握目录的操作方法
- 掌握文件的上传方法
- 掌握访客计算器的制作方法

11.1　文件操作

在不使用数据库系统的情况下，数据可以通过文件（file）来实现数据的存储和读取。这个数据存取的过程也是 PHP 处理文件的过程。这里涉及的文件是文本文件（text file）。

11.1.1　文件数据的写入

对于一个文件的“读”或“写”操作，基本步骤如下：

01 打开文件。
02 从文件里读取数据，或者向文件内写入数据。
03 关闭文件。

打开文件的前提是，文件首先是存在的。如果不存在，则需要建立一个文件，并且在所在的系统环境中，代码应该对文件具有“读”或“写”的权限。

以下实例介绍 PHP 如何处理文件数据。在这个实例中需要把客人订房填写的信息保存到文件中，以便以后使用。

【例 11.1】（实例文件：ch11\11.1.php 和 11.1.1.php）

01 在 PHP 文件同目录下建立一个名称为 booked.txt 的文本文件，然后创建 11.1.php，写入代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/ DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<HEAD><meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/><h2>GoodHome 在线订房表（文件存储）。</h2></HEAD>
<BODY>
<form action="11.2.php" method="post">
<table>
<tr bgcolor="#3399FF" >
    <td>客户姓名:</td>
    <td><input type="text" name="customername" size="20" /></td>
</tr>
<tr bgcolor="#CCCCCC" >
    <td>客户性别：</td>
    <td>
     <select name="gender">
        <option value="m">男</option>
        <option value="f">女</option>
     </select>
  </td>
 </tr>
<tr bgcolor="#3399FF" >
    <td>到达时间:</td>
    <td>
     <select name="arrivaltime">
        <option value="1">一天后</option>
        <option value="2">两天后</option>
        <option value="3">三天后</option>
        <option value="4">四天后</option>
        <option value="5">五天后</option>
     </select>
 </td>
 </tr>
<tr bgcolor="#CCCCCC" >
    <td>电话:</td>
    <td><input type="text" name="phone" size="20" /></td>
</tr>
<tr bgcolor="#3399FF" >
    <td>email:</td>
    <td><input type="text" name="email" size="30" /></td>
</tr>
<tr bgcolor="#666666" >
    <td align="center"><input type="submit" value="确认订房信息" /></td>
</tr>
</table>
</form>
</BODY>
</HTML>
```

02 在 11.1.php 文件的同目录下创建 11.2.php 文件，代码如下。

```
<html>
<head>
```

```
    <title> </title>
    </head>
    <body>
   <?php
    $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
    $customername = trim($_POST['customername']);
    $gender = $_POST['gender'];
    $arrivaltime = $_POST['arrivaltime'];
    $phone = trim($_POST['phone']);
    $email = trim($_POST['email']);

     if( $gender == "m"){
       $customer = "先生";
    }else{
       $customer = "女士";
    }

    $date = date("H:i:s Y m d");
    $string_to_be_added  =  $date."\t".$customername."\t".$customer." 将 在 ".$arrivaltime." 天后到达\t 联系电话: ".$phone."\t Email: ".$email ."\n";
    $fp = fopen("$DOCUMENT_ROOT/booked.txt",'ab');
    if(fwrite($fp, $string_to_be_added, strlen($string_to_be_added))){
      echo $customername."\t".$customer." ,您的订房信息已经保存。我们会通过 email 和电话和您联系。";
    }else{
        echo "信息存储出现错误。";
    }
    fclose($fp);
   ?>
    </body>
    </html>
```

03 运行 11.1.php 文件，最终效果如图 11-1 所示。

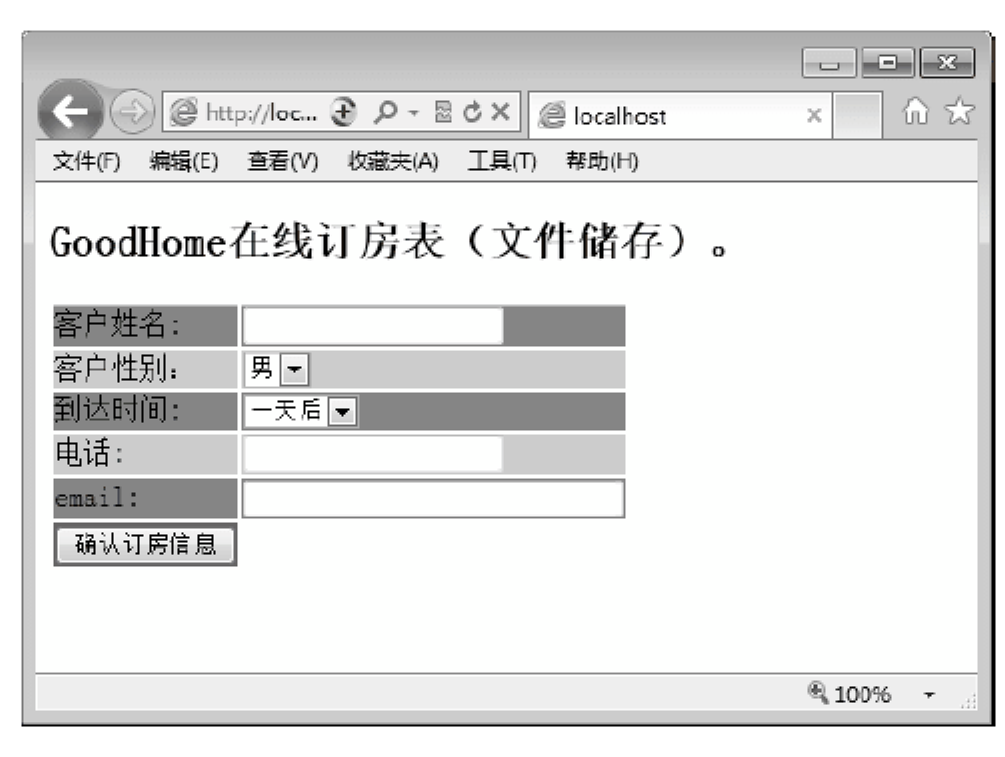


图 11-1 程序运行效果

04 在表单中输入数据，【客户姓名】为“李莉莉”、【到达时间】为“三天后”、【电话】为“159XXXXX266”。单击【确认订房信息】按钮，浏览器会自动跳转到 formfilehandler.php 页

面，并且同时会把数据写入 booked.txt。如果之前没有创建 booked.txt 文件，PHP 会自动创建。运行结果如图 11-2 所示。



图 11-2 程序运行结果

连续写入几次不同的数据，都会被保存到 booked.txt 中。用写字板打开 booked.txt，运行结果如图 11-3 所示。



图 11-3 打开 booked.txt

【案例分析】：

（1）其中，$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT']；通过使用超全局数组 $_SERVER 来确定本系统文件根目录。在 Windows 桌面开发环境中的目录是 c:/wamp/www/。

（2）$customername、$arrivaltime、$phone 为 form4file.html 通过 POST 方法给 formfilehandler.php 传递的数据。

（3）$date 为用 date()函数处理的写入信息时的系统时间。

（4）$string_to_be_added 是要写入 booked.txt 文件的字符串数据。它的格式是通过“\t”和“\n”完成的。“\t”是 tab，“\n”是换新行。

（5）$fp = fopen("$DOCUMENT_ROOT/booked.txt",'ab'); 是 fopen( )函数打开文件并赋值给变量$fp。fopen( )函数的格式是 fopen("Path", "Parameter")。其中，"$DOCUMENT_ROOT/booked.txt" 就是路径（Path），而'ab'是参数（Parameter）。'ab'中的 a 是指在原有文件上继续写入数据，b 则是规定了写入的数据是二进制（binary）的数据模式。

（6）fwrite($fp, $string_to_be_added, strlen($string_to_be_added));是对已经打开的文件进行写入操作。strlen($string_to_be_added)是通过 strlen()函数给出所要写入字符串数据的长度。

（7）在写入操作完成之后，用 fclose()函数关闭文件。

11.1.2 文件数据的读取

到目前为止，数据写入到了文件中，而且文件也可以直接被打开，以查看数据，并对数据进行其他操作。但是，学习 PHP 的一个重要目的，是要通过浏览器对数据进行读取和使用。那么如何读取数据并且通过浏览器进行展示呢？

下面通过实例对文件数据的读取进行了解。

【例 11.2】（实例文件：ch11\11.3.php）

```
<?php
//确认文件路径
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
//确认文件是否存在
 @$fp = fopen("$DOCUMENT_ROOT/booked.txt",'rb');
 if(!$fp){
    echo "没有订房信息。";
  exit;
 }
//循环输出文件内容
 while (!feof($fp)){
    $order = fgets($fp, 2048);
    echo $order. "<br />";
 }
 fclose($fp);     //关闭文件
?>
```

运行结果如图 11-4 所示。

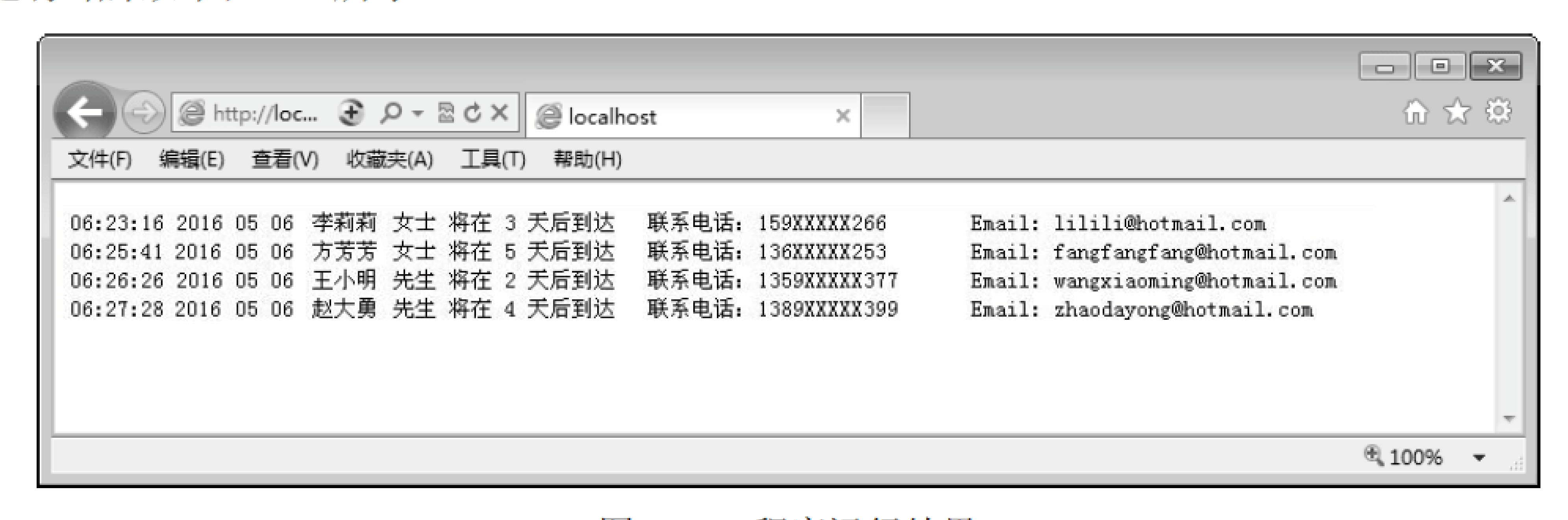


图 11-4 程序运行结果

【案例分析】：

（1）$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];确认文件位置。

（2）fopen()通过参数 rb 打开 booked.txt 文件进行二进制读取。读取内容赋值给变量$fp。$fp 前的@符号用来排除错误提示。

（3）if 语句表示，如果变量$fp 为空，则显示“没有订房信息。”且退出。

（4）While 循环中，!feof($fp)表示只要不到文件尾，就继续 while 循环。循环中 fgets()读取变量$fp 中的内容并赋值给$order。

（5）fgets()中 2048 的参数表示允许读取的最长字节数为 2048-1=2047 字节。

（6）最后 fclose()关闭文件。

不管是读文件还是写文件，其实文件在用 fopen 打开的时候就确定了文件模式，即打开某个特定的文件是用来做什么的。Fopen()中的参数及其用途如表 11-1 所示。

表 11-1　fopen()中的参数及其用途

参数	意义	说明
r	读取	打开文件用于读取，且从文件头开始读取
r+	读取	打开文件用于读取和写入，且从文件头开始读取和写入
w	写入	打开文件用于写入，且从文件头开始写入。如果文件已经存在，则清空原有内容，如果文件不存在，则创建此文件
w+	写入	打开文件用于写入和读取，且从文件头开始写入。如果文件已经存在，则清空原有内容，如果文件不存在，则创建此文件
x	谨慎写入	打开文件用于写入，且从文件头开始写入。如果文件已经存在，则不会被打开，同时 fopen 返回 false，且 PHP 生成警告
x+	谨慎写入	打开文件用于写入和读取，且从文件头开始写入。如果文件已经存在，则不会被打开，同时 fopen 返回 false，且 PHP 生成警告
a	添加	打开文件仅用于添加写入，且在已存在内容之后写入。如果文件不存在，则创建此文件
a+	添加	打开文件用于添加写入和读取，且在已存在内容之后写入。如果文件不存在，则创建此文件
另外两个文件模式		
b	二进制（binary）	配合以上的不同参数使用。二进制文件模式不管是在 Linux 或是 Windows 下都是可使用的。一般情况下，都选择二进制模式
t	文本（text）	文本模式只能在 Windows 下被使用

11.2　目录操作

在 PHP 中，利用相关函数可以实现对目录的操作。常见目录操作函数的使用方法和技巧如下。

1. string getcwd (void)

该函数主要用于获取当前的工作目录，返回的是字符串。下面举例说明此函数的使用方法。

【例 11.3】（实例文件：ch11\11.4.php）

```
<html>
<head>
<title> 获取当前工作目录</title>
</head>
<body>
  <?php
     $d1=getcwd();     //获取当前路径
     echo getcwd();     //输出当前目录
?>
</body>
</html>
```

运行结果如图 11-5 所示。



图 11-5 程序运行效果

2. array scandir (string directory, [int sorting_order])

返回一个 array，包含 directory 中的文件和目录。如果 directory 不是一个目录，则返回布尔值 FALSE，并产生一条 E_WARNING 级别的错误。默认情况下，返回值是按照字母顺序升序排列的。如果使用了可选参数 sorting_order（设为 1），则按照字母顺序降序排列。

下面举例说明此函数的使用方法。

【例 11.4】（实例文件：ch11\11.5.php）

```
<html>
<head>
<title> 获取当前工作目录中的文件和目录</title>
</head>
<body>
  <?php
$dir='d:/ch11';   //定义指定的目录
$files1 = scandir($dir);  //列出指定目录中的文件和目录
$files2 = scandir($dir, 1);
print_r($files1);   //输出指定目录中的文件和目录
print_r($files2);
  ?>
</body>
</html>
```

运行结果如图 11-6 所示。

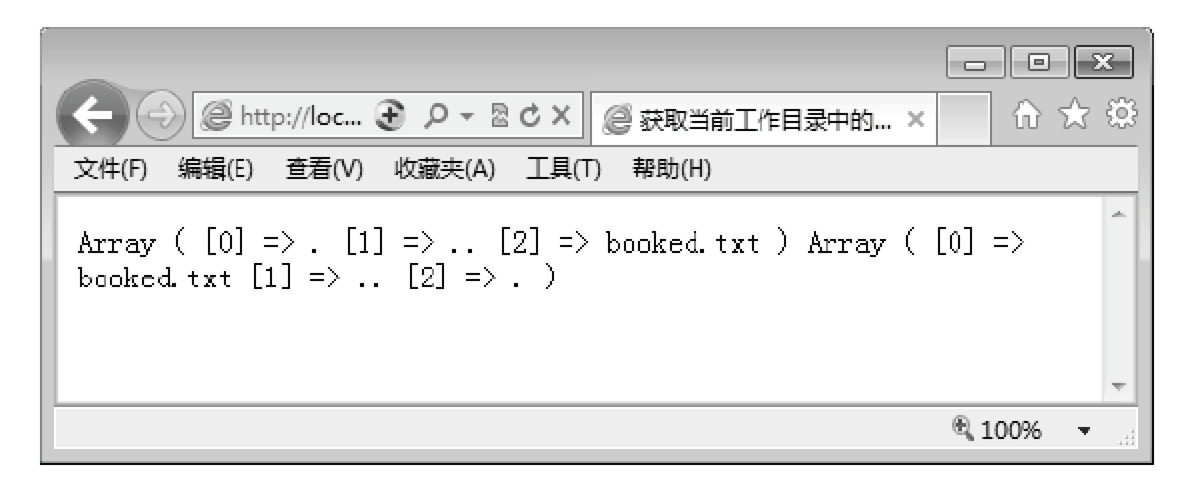


图 11-6 程序运行效果

3. new dir(sting directory)

此函数模仿面向对象机制，将指定的目录名转换为一个对象并返回。使用说明如下。

```
class dir {
dir ( string directory)
string path
resource handle
string read ( void )
void rewind ( void )
void close ( void )
}
```

其中 handle 属性含义为目录句柄、path 属性的含义为打开目录的路径、函数 read(void)含义为读取目录、函数 rewind (void)含义为复位目录、函数 close (void)含义为关闭目录。

下面通过实例说明此函数的使用方法。

【例 11.5】（实例文件：ch11\11.6.php）

```
<html>
<head>
<title> 将目录转换为对象</title>
</head>
<body>
  <?php
$d2 = dir("d:/ch11");                    // 定义目录
echo "Handle: ".$d2->handle."<br />\n";  // 输出目录句柄
echo "Path: ".$d2->path."<br />\n";      // 输出目录的路径
while (false !== ($entry = $d2->read())) {
    echo $entry."<br />\n";
}
$d2->close();
  ?>
</body>
</html>
```

运行结果如图 11-7 所示。

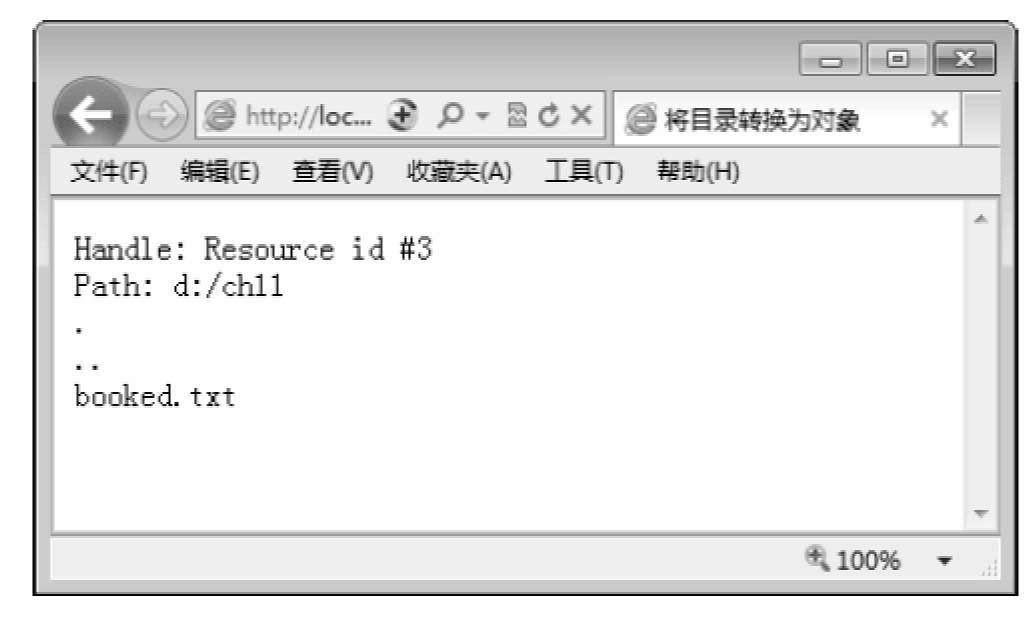


图 11-7 程序运行效果

4. chdir（string directory）

此函数将 PHP 的当前目录改为 directory。如果成功则返回 TRUE，失败则返回 FALSE。下面举例说明此函数的使用方法。

【例 11.6】（实例文件：ch11\11.7.php）

```
<html>
<head>
<title> 将当前目录修改 directory</title>
</head>
<body>
<?php
  if(chdir("d:/ch11")){
      echo "当前目录更改为：d:/ch11<br />";
      }else{
          echo "当前目录更改失败了";
      }
  ?>
</body>
</html>
```

运行结果如图 11-8 所示。

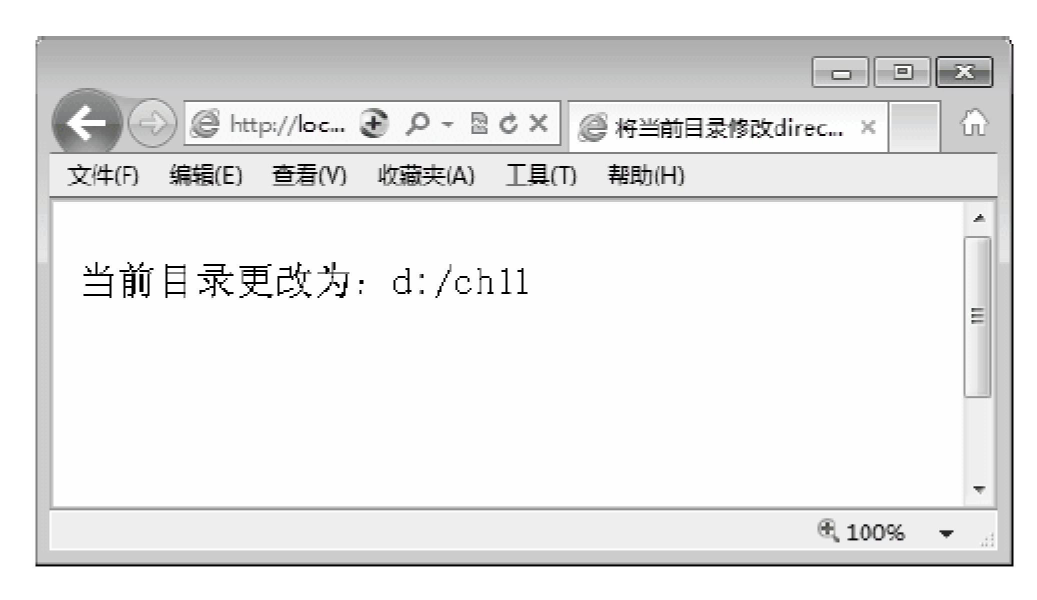


图 11-8　程序运行效果

5. void closedir (resource dir_handle)

此函数主要是关闭由 dir_handle 指定的目录流，另外目录流必须之前被 opendir()所打开。

6. resource opendir(string path)

此函数返回一个目录句柄，其中 path 为要打开的目录路径。如果 path 不是一个合法的目录或者因为权限限制或文件系统错误而不能打开目录，则返回 FALSE 并产生一个 E_WARNING 级别的 PHP 错误信息。如果不想输出错误，可以在 opendir()前面加上“@”符号。

【例 11.7】（实例文件：ch11\11.8.php）

```
<html>
<head>
<title> </title>
</head>
<body>
<?php
$dir = "d:/ch11/";
// 打开一个目录，然后读取目录中的内容
if (is_dir($dir)) {
    if ($dh = opendir($dir)) {
```

```
        while (($file = readdir($dh)) !== false) {
            print "filename: $file : filetype: " . filetype($dir . $file) . "\n";
        }
        closedir($dh);
    }
}
?>
</body>
</html>
```

运行结果如图 11-9 所示。

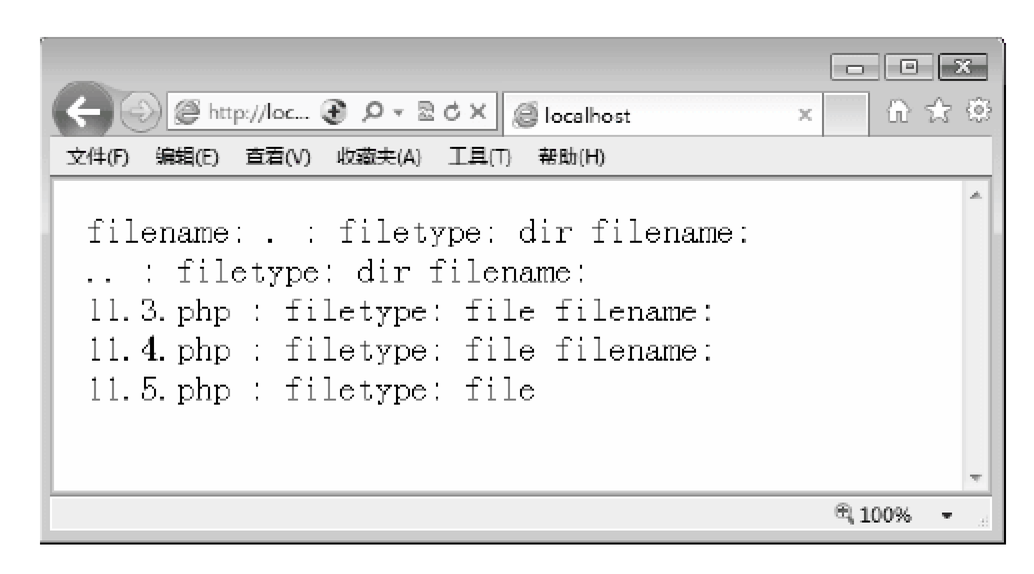


图 11-9　程序运行效果

其中 is_dir()函数主要判断给定文件名是否是一个目录，readdir()函数从目录句柄中读取条目，closedir()函数关闭目录句柄。

7. string readdir (resource dir_handle)

该函数主要是返回目录中下一个文件的文件名。文件名以在文件系统中的排序返回。

【例 11.8】（实例文件：ch11\11.9.php）

```
<html>
<head>
<title> </title>
</head>
<body>
<?php
// 注意在 4.0.0-RC2 之前不存在“!==”运算符
if ($handle = opendir('d:/ch11')) {
    echo "Directory handle: $handle\n";
    echo "Files:\n";
    /* 这是正确遍历的目录方法 */
    while (false !== ($file = readdir($handle))) {
        echo "$file\n";
    }
    closedir($handle);
}
?>
</body>
```

```
</html>
```

运行结果如图 11-10 所示。

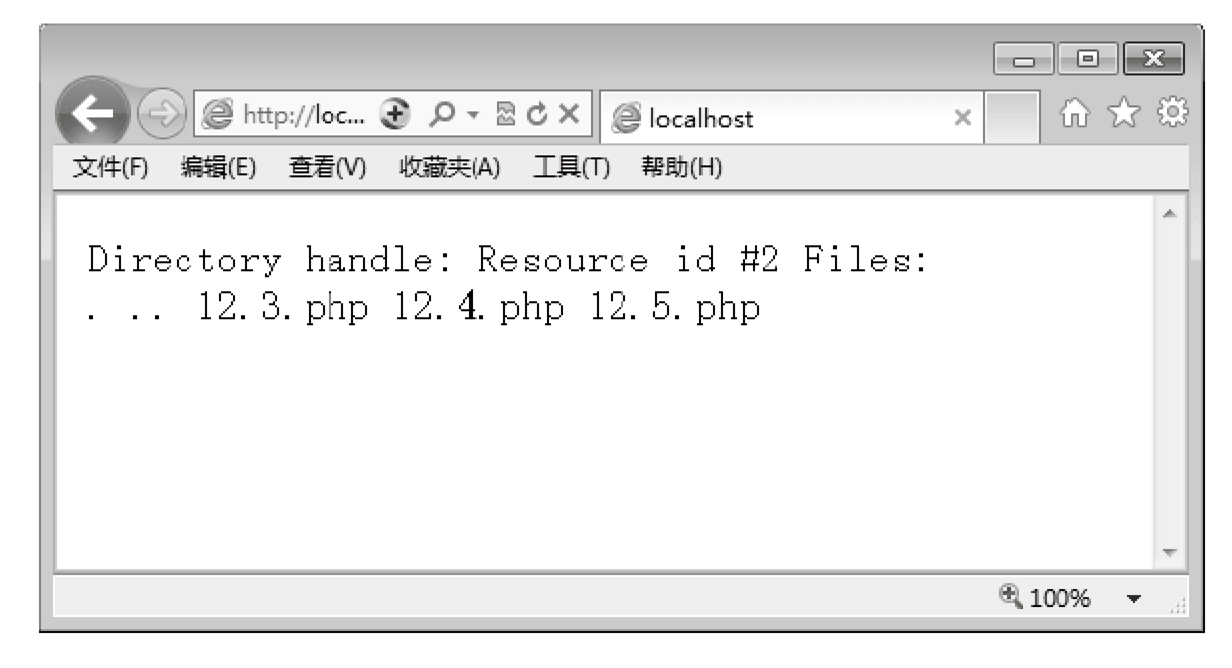


图 11-10 程序运行效果

在遍历目录时，有的读者会经常写出下面错误的遍历方法。

```
/* 这是错误的遍历目录的方法 */
while ($file = readdir($handle)) {
    echo "$file\n";
}
```

11.3 文件的上传

在网络中用户可以上传自己的文件。实现这种功能的方法有很多，用户把一个文件上传到服务器，需要在客户端和服务器端建立一个通道传递文件的字节流，并在服务器中进行上传操作。下面介绍一种代码最少并且容易理解的方法。

下面的实例主要讲述如何实现文件的上传功能，具体操作步骤如下。

【例 11.9】（实例文件：ch11\11.10.php 和 11.10.1.php）

01 首先创建一个实现文件上传功能的文件。为了设置保存上传文件的路径，用户需要在创建文件的目录下新建一个名称为 file 的文件夹。然后新建 11.10.1.php 文件，代码如下。

```
<html>
<head>
<title>实现上传文件</title>
</head>
<body>
<?php
if ($_POST[add]=="上传"){
      //根据现在的时间产生一个随机数
      $rand1=rand(0,9);
      $rand2=rand(0,9);
      $rand3=rand(0,9);
      $filename=date("Ymdhms").$rand1.$rand2.$rand3;
      if(empty($_FILES['file_name']['name'])){
       //$_FILES['file_name']['name']为获取客户端机器文件的原名称
```

```
            echo "文件名不能为空";
            exit;
            }
         $oldfilename=$_FILES['file_name']['name'];
         echo "<br />原文件名为: ".$oldfilename;

$filetype=substr($oldfilename,strrpos($oldfilename,"."),strlen($oldfilename)-s
trrpos($oldfilename,"."));
         echo "<br />原文件的类型为: ".$filetype;

if(($filetype!='.doc')&&($filetype!='.xls')&&($filetype!='.DOC')&&($filetype!=
'.XLS')){
            echo "<script>alert('文件类型或地址错误');</script>";
            echo "<script>location.href='11.3.php';</script>";
            exit;
            }
         echo "<br />上传文件的大小为(字节): ".$_FILES['file_name']['size'];
      //$_FILES['file_name']['size']为获取客户端机器文件的大小，单位为 B
         if ($_FILES['file_name']['size']>1000000) {
            echo "<script>alert('文件太大，不能上传');</script>";
            echo "<script>location.href='11.3.php';</script>";
            exit;
            }
         echo  "<br  /> 文 件 上 传 服 务 器 后 的 临 时 文 件 名 为 :
".$_FILES['file_name']['tmp_name'];
      //取得保存文件的临时文件名(含路径)
         $filename=$filename.$filetype;
         echo "<br />新文件名为: ".$filename;
         $savedir="file/".$filename;
         if(move_uploaded_file($_FILES['file_name']['tmp_name'],$savedir)){
             $file_name=basename($savedir);      //取得保存文件的文件名(不含路径)
             echo "<br />文件上传成功! 保存为: ".$savedir;
            }else{
             echo "<script language=javascript>";
             echo "alert('错误，无法将附件写入服务器!\n 本次发布失败! ');";
             echo "location.href='11.3.php?';";
             echo "</script>";
             exit;
           }
      }
   ?>
   </body>
   </html>
```

代码分析如下：

（1）需要首先创建变量，设定文件的上传类型、保存路径和程序所在路径。

（2）实现自定义函数获取文件后缀名和生成随机文件名。在上传过程中，如果上传了大量的文件，可能会出现文件名称重复的现象，所以本实例在文件上传的过程中，首先获取上传文件的后缀名称并结合随机产生的数字，生成一个新的文件，避免了文件名称重复的现象。

（3）判断获取的文件类型是否符合指定类型，如果文件名称符合，则为给该文件生成一个具有随机性质的名称，并使用 move_uploaded_file 函数完成文件的上传，否则显示提示信息。

02 下面创建一个获取上传文件的页面。创建文件 11.10.php，代码如下。

```
<html>
<head>
<title>上传文件</title>
</head>
<h3 align="center">上传文件</h3>
<form method="post" action="11.10.1.php" enctype="multipart/form-data">
    <table border=0 cellspacing=0 cellpadding=0 align=center width="100%">
     <tr>
           <td height="16">
           <input name="file" type="file"  value="浏览" >
           <input type="submit" value="上传" name="B1">
           </td>
     </tr>
    </table>
    </form>
</body>
</html>
```

其中“<form method="post" action="11.10.1.php" enctype="multipart/form-data">”语句中 method 属性表示提交信息的方式是 post，即采用数据块，action 属性表示处理信息的页面为 11.10.1.php，enctype="multipart/form-data"表示以二进制的方式传递提交的数据。

运行结果如图 11-11 所示。单击【浏览】按钮，即可选择需要上传的文件，最后单击【上传】按钮即可实现上传操作。



图 11-11　程序运行结果

11.4 实战演练——编写文本类型的访客计算器

下面通过对文本文件的操作，利用相关函数编写一个简单的文本类型的访客计算器。

【例 11.10】（实例文件：ch11\11.11.php）

```
<html>
     <head>
         <title>访客计数器</title>
     </head>
     <body>
        <?php
            if (!@$fp=fopen("coun.txt","r")){
                 //只读方式打开 coun.txt 文件
                 echo "coun.txt 文件创建成功! <br />";
               }
            @$num=fgets($fp,12);   //读取11位数字
            if ($num=="") $num=0;
               //如果文件的内容为空，初始化为0
            $num++;              //浏览次数加一
            @fclose($fp);          //关闭文件
            $fp=fopen("coun.txt", "w");//只写方式打开 coun.txt 文件
            fwrite($fp,$num);    //写入加一后结果
            fclose($fp);         //关闭文件
            echo "您是第".$num."位浏览者!";        //浏览器输出浏览次数
        ?>
    </body>
</html>
```

程序第一次运行的结果如图 11-12 所示。

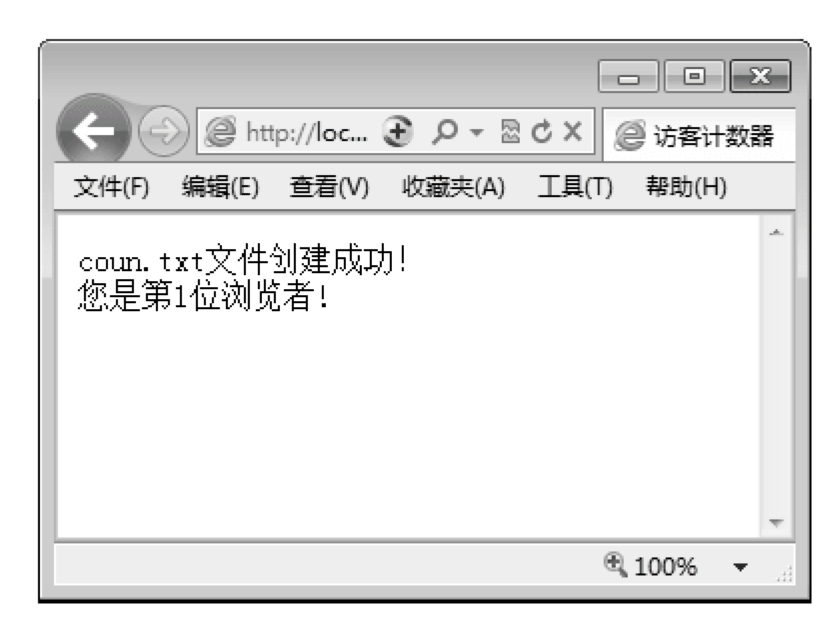


图 11-12 程序运行效果

由结果可以看出，该程序首先创建一个 count.txt 的文本文件，用于保存浏览次数。首先打开这个文件，然后初始化数据为 0，并实现加 1 操作。

11.5　高手私房菜

技巧 1：如何批量上传多个文件？

本章讲述了如何上传单个文件，那么如何上传多个文件呢？用户只需要在表单中使用复选框以数组式提交语法即可。

提交的表单语句如下。

```
<form method="post" action="11.3.1.php" enctype="multipart/form-data">
    <table border=0 cellspacing=0 cellpadding=0 align=center width="100%">
      <input name="userfile[]" type="file"  value="浏览 1" >
      <input name=" userfile[]" type="file"  value="浏览 2" >
      <input name="f userfile[]" type="file"  value="浏览 3" >
      <input type="submit" value="上传" name="B1">
    </table>
</form>
```

技巧 2：如何从文件中读取一行？

在 PHP 网站开发中，支持从文件指针中读取一行。使用 string fgets(int handle, [int length])函数即可实现上述功能。其中 int handle 是要读入数据的文件流指针，fopen()函数返回数值；int length 设置读取的字符个数，读入的字符个数为 length-1。如果没有指定 length，则默认为 1024 个字节。

11.6　经典习题

（1）　制作一个包含读取指定文件的例子。

（2）　制作一个包含写入数据到文件的例子。

（3）　制作一个读取文件目录的例子。

（4）　制作一个图片文件上传的例子。

（5）　制作一个记录访客计算器的例子。

第 12 章　图形图像处理

PHP 不仅可以输出纯 HTML，还可以创建及操作多种不同图像格式的图像文件，包括 gif、png、jpg、wbmp 和 xpm 等。更方便的是，PHP 可以直接将图像流输出到浏览器。要处理图像，需要在编译 PHP 时加上图像函数的 GD 库，另外还可以使用第三方的图形库。本章将讲述图形图像的处理方法和技巧。

本章学习目标

- 掌握在 PHP 中加载 GD 库的方法
- 掌握图像图像创建的基本方法
- 掌握 Jpgraph 库的使用方法
- 掌握 3D 饼形图的制作方法

12.1　在 PHP 中加载 GD 库

PHP 中的图形图像处理功能，都要求有一个库文件的支持，这就是 GD2 库。PHP 5 自带此库。

如果在 Windows 7 系统环境下，修改 php.ini 中 extension=php_gd2.dll 前面的“；”即可启用，如图 12-1 所示。

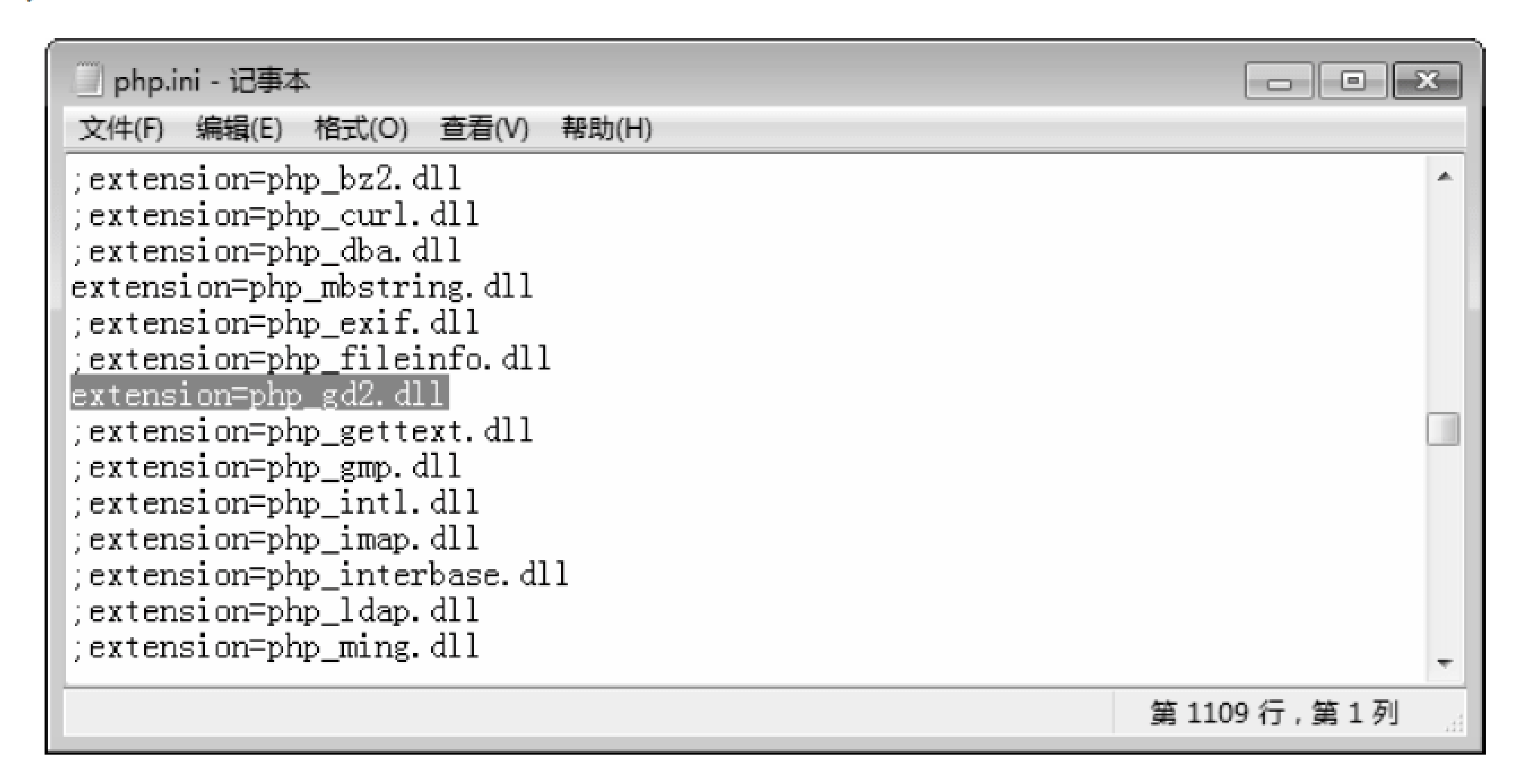


图 12-1　修改 php.ini 配置文件

下面了解一下 PHP 中常用图像函数的功能，如表 12-1 所示。

表 12-1　图像函数的功能

函数	功能
gd_info	取得当前安装的 GD 库的信息
getimagesize	取得图像大小
image_type_to_mime_type	取得 getimagesize、exif_read_data、exif_thumbnail、exif_imagetype 所返回的图像类型的 MIME 类型
image2wbmp	以 WBMP 格式将图像输出到浏览器或文件
imagealphablending	设定图像的混色模式
imageantialias	是否使用 antialias 功能
imagearc	画椭圆弧
imagechar	水平地画一个字符
imagecharup	垂直地画一个字符
imagecolorallocate	为一幅图像分配颜色
imagecolorallocatealpha	为一幅图像分配颜色
imagecolorat	取得某像素的颜色索引值
imagecolorclosest	取得与指定的颜色最接近的颜色的索引值
imagecolorclosestalpha	取得与指定的颜色最接近的颜色
imagecolorclosesthwb	取得与给定颜色最接近的色度的黑白色的索引
imagecolordeallocate	取消图像颜色的分配
imagecolorexact	取得指定颜色的索引值
imagecolorexactalpha	取得指定的颜色的索引值
imagecolormatch	使一个图像中调色板版本的颜色与真彩色版本更能匹配
imagecolorresolve	取得指定颜色的索引值或有可能得到的最接近的替代值
imagecolorresolvealpha	取得指定颜色的索引值或有可能得到的最接近的替代值
imagecolorset	给指定调色板索引设定颜色
imagecolorsforindex	取得某索引的颜色
imagecolorstotal	取得一幅图像的调色板中颜色的数目
imagecolortransparent	将某个颜色定义为透明色
imagecopy	拷贝图像的一部分
imagecopymerge	拷贝并合并图像的一部分
imagecopymergegray	用灰度拷贝并合并图像的一部分
imagecopyresampled	重采样拷贝部分图像并调整大小
imagecopyresized	拷贝部分图像并调整大小
imagecreate	新建一个基于调色板的图像
imagecreatefromgd2	从 GD2 文件或 URL 新建一个图像
imagecreatefromgd2part	从给定的 GD2 文件或 URL 中的部分新建一个图像
imagecreatefromgd	从 GD 文件或 UR 新建一个图像
imagecreatefromgif	从 GIF 文件或 URL 新建一个图像
imagecreatefromjpeg	从 JPEG 文件或 URL 新建一个图像
imagecreatefrompng	从 PNG 文件或 URL 新建一个图像
imagecreatefromstring	从字符串中的图像流新建一个图像
imagecreatefromwbmp	从 WBMP 文件或 URL 新建一个图像
imagecreatefromxbm	从 XBM 文件或 URL 新建一个图像
imagecreatefromxpm	从 XPM 文件或 URL 新建一个图像
imagecreatetruecolor	新建一个真彩色图像
imagedashedline	画一个虚线
imagedestroy	销毁一个图像
imageellipse	画一个椭圆

（续表）

函数	功能
Imagefill	区域填充
imagefilledarc	画一个椭圆弧且填充
imagefilledellipse	画一个椭圆并填充
imagefilledpolygon	画一个多边形并填充
imagefilledrectangle	画一个矩形并填充
imagefilltoborder	区域填充到指定颜色的边界为止
imagefontheight	取得字体高度
imagefontwidth	取得字体宽度
imageftbbox	取得使用 FreeType 2 字体的文本的范围
imagefttext	使用 FreeType 2 字体将文本写入图像
imagegd	将 GD 图像输出到浏览器或文件
imagegif	以 GIF 格式将图像输出到浏览器或文件
imagejpeg	以 JPEG 格式将图像输出到浏览器或文件
imageline	画一条直线
imagepng	将调色板从一幅图像拷贝到另一幅
imagepolygon	画一个多边形
imagerectangle	画一个矩形
imagerotate	用给定角度旋转图像
imagesetstyle	设定画线的风格
imagesetthickness	设定画线的宽度
imagesx	取得图像宽度
imagesy	取得图像高度
imagetruecolortopalette	将真彩色图像转换为调色板图像
imagettfbbox	取得使用 TrueType 字体的文本的范围
imagettftext	用 TrueType 字体向图像写入文本

12.2 图形图像的典型应用案例

下面讲述图形图像的经典使用案例。

12.2.1 创建一个简单的图像

使用 GD2 库文件，就像使用其他库文件一样。由于它是 PHP 的内置库文件，不需要在 PHP 文件中再用 include 等函数进行调用。以下实例介绍图像的创建方法。

【例 12.1】（实例文件：ch12\12.1.php）

```
<?php
   $im = imagecreate(200,300);                        //创建一个画布
   $white = imagecolorallocate($im, 8,2,133);            //设置画布的背景颜色为蓝色
   imagegif($im);                                  //输出图像
?>
```

运行程序，结果如图 12-2 所示。本实例使用 imagecreate()函数创建了一个宽 200 像素、高 300 像素的画布，并设置画布的 RGB 值为（8，2，133），最后输出一个 gif 格式的图像。

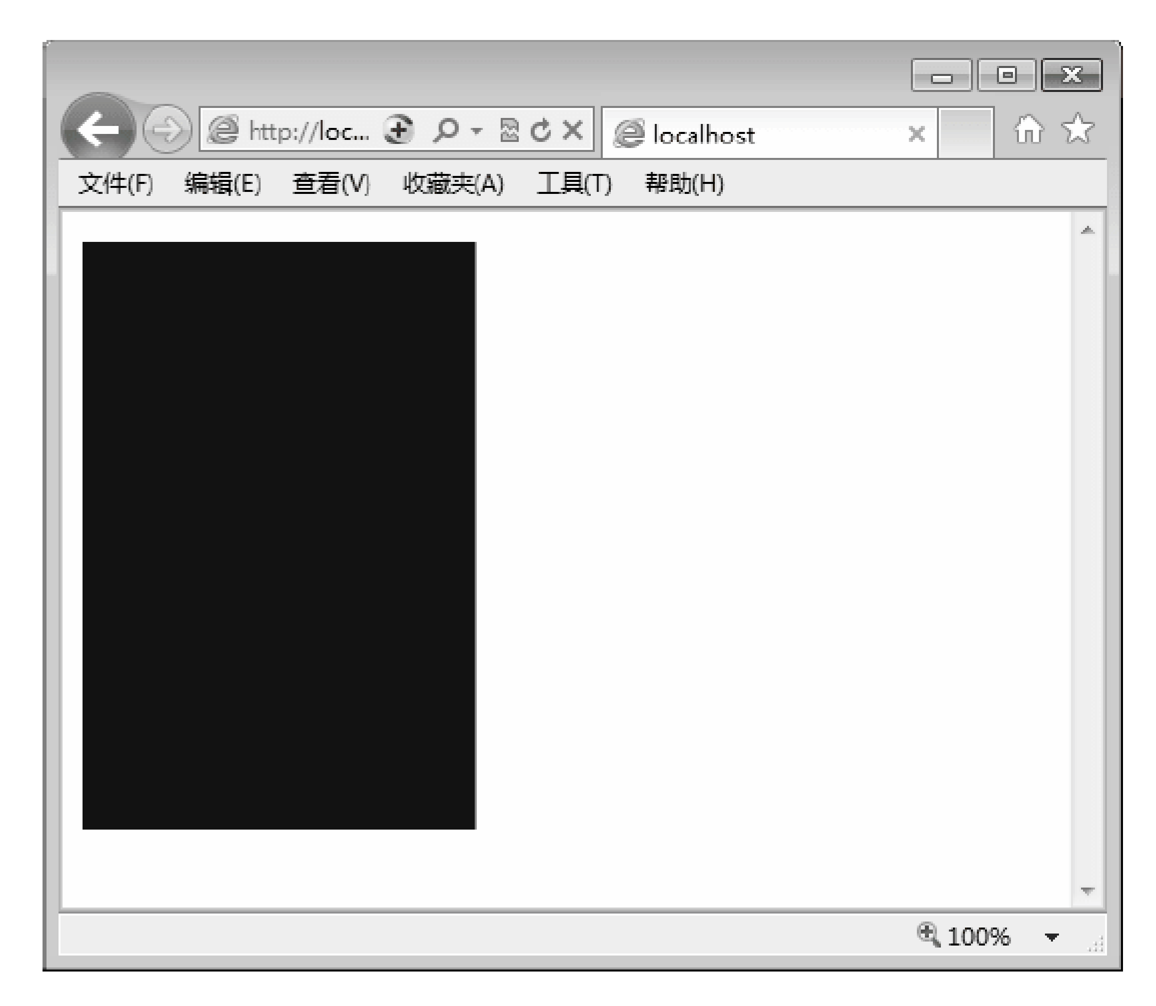


图 12-2　程序运行效果



提　示

使用 imagecreate(200,300)函数创建基于普通调色板的画布，支持 256 色，其中 200、300 为图像的宽度和高度，单位为像素。

上面的实例只是把图片输出到页面，那么如何保存需要的图片文件呢？下面通过实例介绍图像文件的创建方法。

【例 12.2】（实例文件：ch12\12.2.php）

```
<?php
   $ysize =200;
   $xsize =300;
   $theimage = imagecreatetruecolor($xsize,$ysize);        //创建图片画布大小
   $color2 = imagecolorallocate($theimage, 8,2,133);       //定义颜色 color2
   $color3 = imagecolorallocate($theimage, 230,22,22);     //定义颜色 color3
   imagefill($theimage, 0, 0,$color2);
   imagearc($theimage,100,100,150,200,0,270,$color3);     //创建一个弧线
   imagejpeg($theimage,"newimage.jpeg");                //生成 jpeg 格式的图片
  //输出 png 格式的图片
   header('content-type: image/png');
   imagepng($theimage);
   imagedestroy($theimage);                            //清除对象，释放资源
?>
```

运行程序，结果如图 12-3 所示。同时在程序文件夹下生成一个名为 newimage.jpeg 的图片，其内容与页面显示相同。

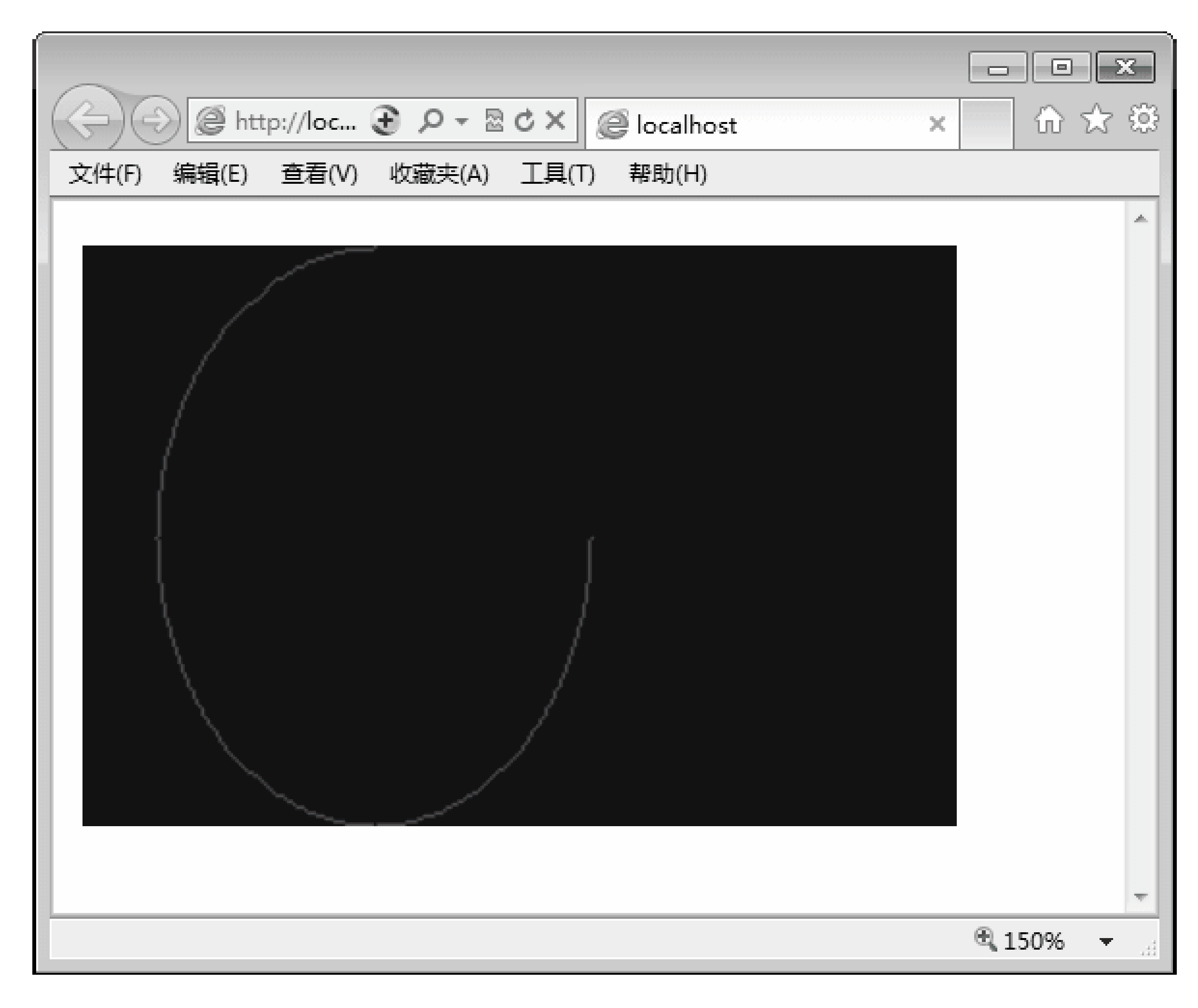


图 12-3　程序运行效果

【案例分析】：

（1）其中，imagecreatetruecolor()函数是用来创建图片画布的。它需要两个参数，一个是 x 轴的大小，一个是 y 轴的大小。$xsize =200;和$ysize =300;分别设定了这两参数的大小。$theimage = imagecreatetruecolor($xsize,$ysize);使用这两个参数生成了画布，并且赋值为$theimage。

（2）imagearc($theimage,100,100,150,200,0,270,$color3);语句使用了 imagearc()函数来在画布上创建一个弧线。它的参数分为以下几个部分：$theimage 为目标画布，“100,100”为弧线中心点的 x、y 坐标，“150,200”为弧线的宽度和高度，“0,270”为顺时针画弧线的起始度数和终点度数，在 0 到 360 度之间，$color3 为画弧线所使用的颜色。

（3）imagejpeg()函数是生成 jpeg 格式的图片的函数。imagejpeg($theimage,"newimage.jpeg");把画布对象$theimage 生成一个名为 newimage.jpeg 的 jpeg 图片文件，并且直接存储在当前路径下。

（4）同时，header('content-type: image/png');和 imagepng($theimage);向页面输出 png 格式的图片。

（5）最后清除对象，释放资源。

12.2.2　使用 GD2 函数在照片上添加文字

上面创建了一个图片。如果想在图片上添加文字，就需要修改图片，具体的过程为先读取一个图片，然后修改这个图片。

【例 12.3】（实例文件：ch12\12.3.php）

```
  <?php
     $theimage = imagecreatefromjpeg('newimage.jpeg');           //读取图像并赋值
给 theimage
     $color1 = imagecolorallocate($theimage, 255,255,255);          //创建颜色
color1
     $color3 = imagecolorallocate($theimage, 230,22,22);            //创建颜色
color3
```

```
    imagestring($theimage,5,60,100,'Text added to this image.',$color1); //向图片添加字符串
   //处理输出到页面的 png 图片
   header('content-type: image/png');
    imagepng($theimage);
    imagepng($theimage,'textimage.png');    //创建 png 格式图片
    imagedestroy($theimage);           //清除对象，释放资源
?>
```

运行程序，结果如图 12-4 所示。同时在程序所在的文件夹下生成一个名为 newimage.jpeg 的图片，其内容与页面显示相同。

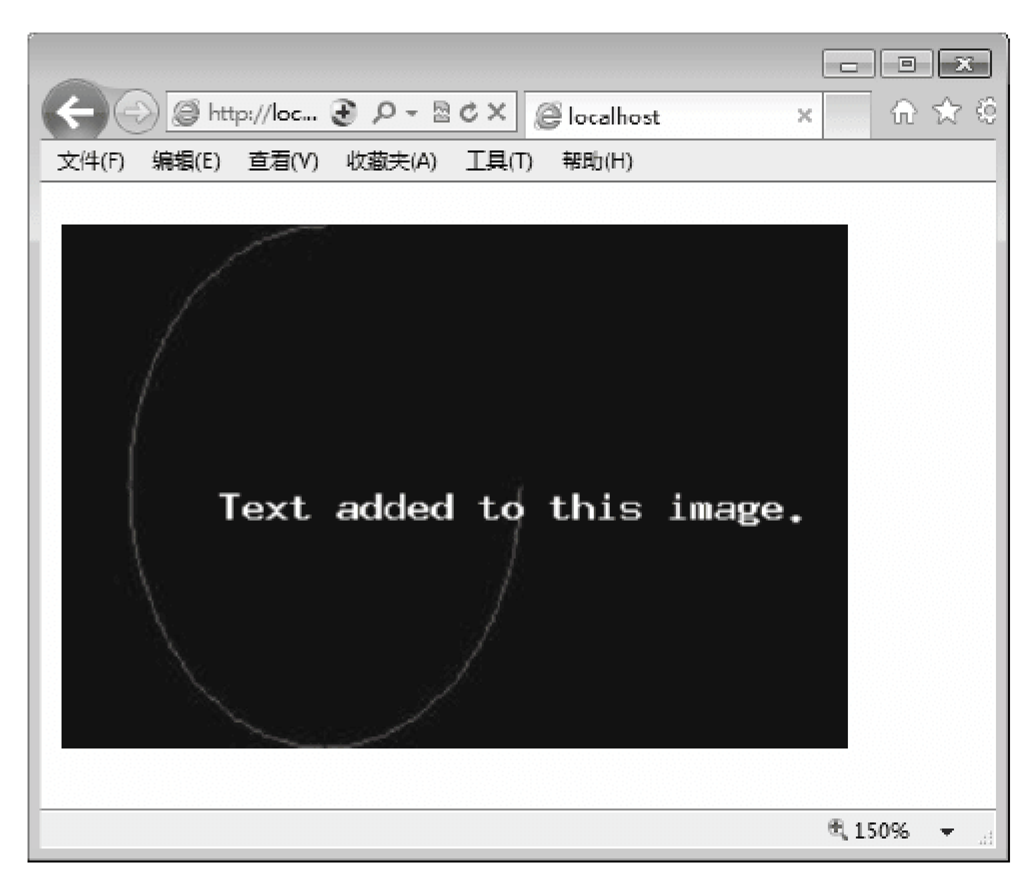


图 12-4 程序运行效果

【案例分析】：

（1）imagecreatefromjpeg('newimage.jpeg');语句中 imagecreatefromjpeg()函数从当前路径下读取 newimage.jpeg 图形文件，并且传递给$theimage 变量作为对象，以待操作。

（2）选取颜色后，imagestring($theimage,5,60,100,'Text added to this image.',$color1);语句中的 imagestring()函数向对象图片添加字符串'Text added to this image.'。这里的参数中，$theimage 为对象图片；“5”为字体类型，这个字体类型的参数从 1 到 5 代表不同的字体；“60,100”为字符串添加的起始 x 与 y 的坐标；'Text added to this image.'为要添加的字符串，现在的情况下只支持 asc 字符；$color1 为写字的颜色。

（3）header('content-type: image/png');和 imagepng($theimage);语句共同处理输出到页面的 png 图片。之后 imagepng($theimage,'textimage.png');语句就创建文件名为 textimage.png 的 png 图片，并保存在当前路径下。

12.2.3 使用 TrueType 字体处理中文生成的图片

字体处理在很大程度上是 PHP 图形处理经常要面对的问题。imagestring()函数默认的字体是十分有限的。这就要进入字体库文件，而 TureType 字体是字体中极其常用的格式。比如在 Windows 下，打开 C:\WINDOWS\Fonts 目录，会出现很多字体文件，其中绝大部分是 TureType 字体，如图 12-5 所示。



图 12-5 系统中的字体

PHP 使用 GD2 库，在 Windows 环境下，需要给出 TureType 字体所在的文件夹路径，如在文件开头使用以下语句：

```
putenv('GDFONTPATH=C:\WINDOWS\Fonts');
```

使用 TureType 字体也可以直接使用 imagettftext()函数，它是使用 ttf 字体的 imagestring()函数，它的格式为：

```
imagettftext（图片对象，字体大小，文字显示角度，起始 x 坐标，起始 y 坐标，文字颜色，字体名称，文字信息）
```

另外一个很重要的问题就是，GD 库中的 imagettftext()函数默认无法支持中文字符，添加到图片上去的。这是因为 GD 库的 imagettftext()函数对于字符的编码是采用的 UTF-8 编码格式，而简体中文的默认格式为 GB2312。

下面就介绍这样的一个例子，具体操作步骤如下。

01 把 C:\WINDOWS\Fonts 下的字体文件 simhei.ttf 复制到和文件 12.4.php 同目录下。

02 在网站目录下建立 12.4.php，输入以下代码并保存。

```
<?php
    $ysize =200;
    $xsize =300;
    $theimage = imagecreatetruecolor($xsize,$ysize);      //创建画布、填充颜色
color2
    $color2 = imagecolorallocate($theimage, 8,2,133);     //定义颜色 color2
    $color3 = imagecolorallocate($theimage, 230,22,22);   //定义颜色 color3
    imagefill($theimage, 0, 0,$color2);                   //给画布填充颜色 color2
    $fontname='simhei.ttf';                               //将黑体字的路径赋值给变量
fontname
    $zhtext = "这是一个把中文用黑体显示的图片。";
    $text = iconv("GB2312", "UTF-8", $zhtext);           //中文编码转换为 UTF-8
    imagettftext($theimage,12,0,20,100,$color3,$fontname,$text); //将字符串写到
```

```
画布上
    header('content-type: image/png');                          //输出为 png 格式
    imagepng($theimage);
    imagedestroy($theimage);                               //清除对象，释放资源
?>
```

运行程序，结果如图 12-6 所示。

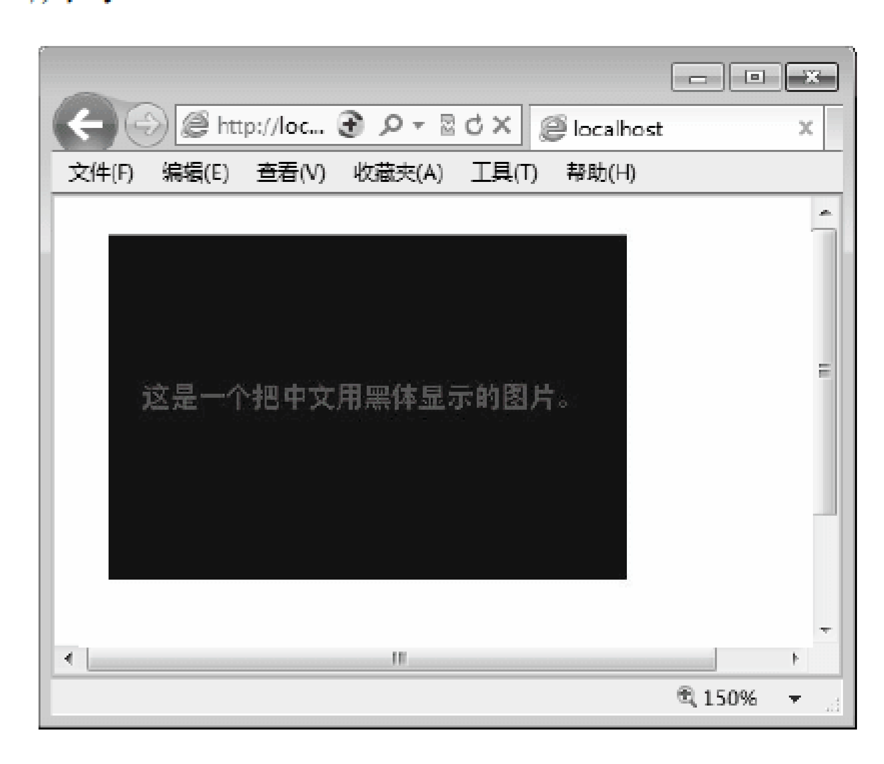


图 12-6　程序运行效果

【案例分析】：

（1）其中 imagefill($theimage, 0, 0,$color2);语句用于创建画布、填充颜色。

（2）$fontname='simhei.ttf';语句确认当前目录下的黑体字的 ttf 文件，并且把路径赋值给$fontname 变量。

（3）$zhtext 中为中文字符，它的编码为 GB2312。为了转换此编码为 UTF-8，使用$text =iconv("GB2312", "UTF-8", $zhtext);语句把$zhtext 中的中文编码转换为 UTF-8，并赋值给$text 变量。

（4）imagettftext($theimage,12,0,20,100,$color3,$fontname,$text);语句按照 imagettftext()函数的格式分别确认了参数。$theimage 为目标图片，“12”为字符的大小，“0”为显示的角度，“20,100”为字符串显示的初始 x 和 y 的值，$fontname 为已经设定的黑体，$text 为已经转换为 UTF-8 格式的中文字符。

12.3　Jpgraph 库的使用

Jpgraph 是一个功能强大且十分流行的 PHP 外部图片处理库文件，它是建立在内部库文件 GD2 库之上的。它的优点是建立了很多方便操作的对象和函数，能够大大简化使用 GD 库对图片进行处理的编程过程。

12.3.1　Jpgraph 的安装

Jpgraph 的安装就是 PHP 对 Jpgraph 类库的调用。可以采用多种形式，但是，首先都需要到 Jpgraph 的官方网站下载类库文件的压缩包，下载的最新压缩包为 Jpgraph3.5.0b1。

解压以后，如果是 Linux 系统，可以把它放置在 lib 目录下，并且使用下面的语句重命名此类库的文件夹。

```
ln -s jpgraph-3.x jpgraph
```

如果是 Windows 系统，在本机 WAMP 环境下，则可以把类库文件夹放在 www 目录下，或者放置在项目的文件夹下。然后在程序中引用的时候，直接使用 require_once()命令，并且指出 Jpgraph 类库相对于此应用的路径。

在本机环境下，把 jpgraph 文件夹放置在 D:\PHP 7\ch12 文件夹下。在应用程序的文件中加载此库，require_once ('jpgraph/src/jpgraph.php');即可。

12.3.2 Jpgraph 的配置

使用 Jpgraph 类之前，需要对 PHP 系统的一些限制性参数进行修改。具体修改以下 3 个方面的内容。

- 需要到 php.ini 中修改内存限制，memory_limit 至少为 32MB，本机环境为 momery_limit = 64m。
- 最大执行时间 max_execution_time 要增加，Jpgraph 类的官方推荐时间为 30 秒，即 max_execution_time =30。
- 用 “;” 号注释掉 output_buffering 选项。

12.3.3 制作柱形与折线统计图

安装设置生效以后，就可以使用此类库了。由于 Jpgraph 有很多实例，所以读者可以轻松地通过实例来学习。

下面通过实例来学习 Jpgraph 类的使用方法和技巧。

01 找到安装过的 jpgraph 类库文件夹，在其下的 src 文件夹下找到 Examples 文件夹。找到 barlinealphaex1.php 文件，将其复制到 ch12 文件夹下。在浏览器中打开，其代码如下。

```
<?php
 // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_bar.php');
require_once ('jpgraph/jpgraph_line.php');
$ydata  = array(10,120,80,190,260,170,60,40,20,230);
$ydata2 = array(10,70,40,120,200,60,80,40,20,5);
$months = $gDateLocale->GetShortMonth();
$graph = new Graph(300,200);
$graph->SetScale("textlin");
$graph->SetMarginColor('white');
$graph->SetMargin(30,1,20,5);
$graph->SetBox();
$graph->SetFrame(false);
$graph->tabtitle->Set('Year 2003');
$graph->tabtitle->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->ygrid->SetFill(true,'#DDDDDD@0.5','#BBBBBB@0.5');
$graph->ygrid->SetLineStyle('dashed');
```

```
$graph->ygrid->SetColor('gray');
$graph->xgrid->Show();
$graph->xgrid->SetLineStyle('dashed');
$graph->xgrid->SetColor('gray');
$graph->xaxis->SetTickLabels($months);
$graph->xaxis->SetFont(FF_ARIAL,FS_NORMAL,8);
$graph->xaxis->SetLabelAngle(45);
$bplot = new BarPlot($ydata);
$bplot->SetWidth(0.6);
$fcol='#440000';
$tcol='#FF9090';
$bplot->SetFillGradient($fcol,$tcol,GRAD_LEFT_REFLECTION);
$bplot->SetWeight(0);
$graph->Add($bplot);
$lplot = new LinePlot($ydata2);
$lplot->SetFillColor('skyblue@0.5');
$lplot->SetColor('navy@0.7');
$lplot->SetBarCenter();
$lplot->mark->SetType(MARK_SQUARE);
$lplot->mark->SetColor('blue@0.5');
$lplot->mark->SetFillColor('lightblue');
$lplot->mark->SetSize(6);
$graph->Add($lplot);
$graph->Stroke();
?>
```

02 修改 require_once ('jpgraph/jpgraph.php');为 require_once ('jpgraph/src/jpgraph.php');。修改 require_once ('jpgraph/jpgraph_bar.php ');为 require_once ('jpgraph/src/jpgraph_bar.php');。修改 require_once ('jpgraph/jpgraph_line.php');为 require_once ('jpgraph/src/jpgraph_line.php');，以载入本机 jpgraph 类库。

03 运行 barlinealphaex1.php，结果如图 12-7 所示。

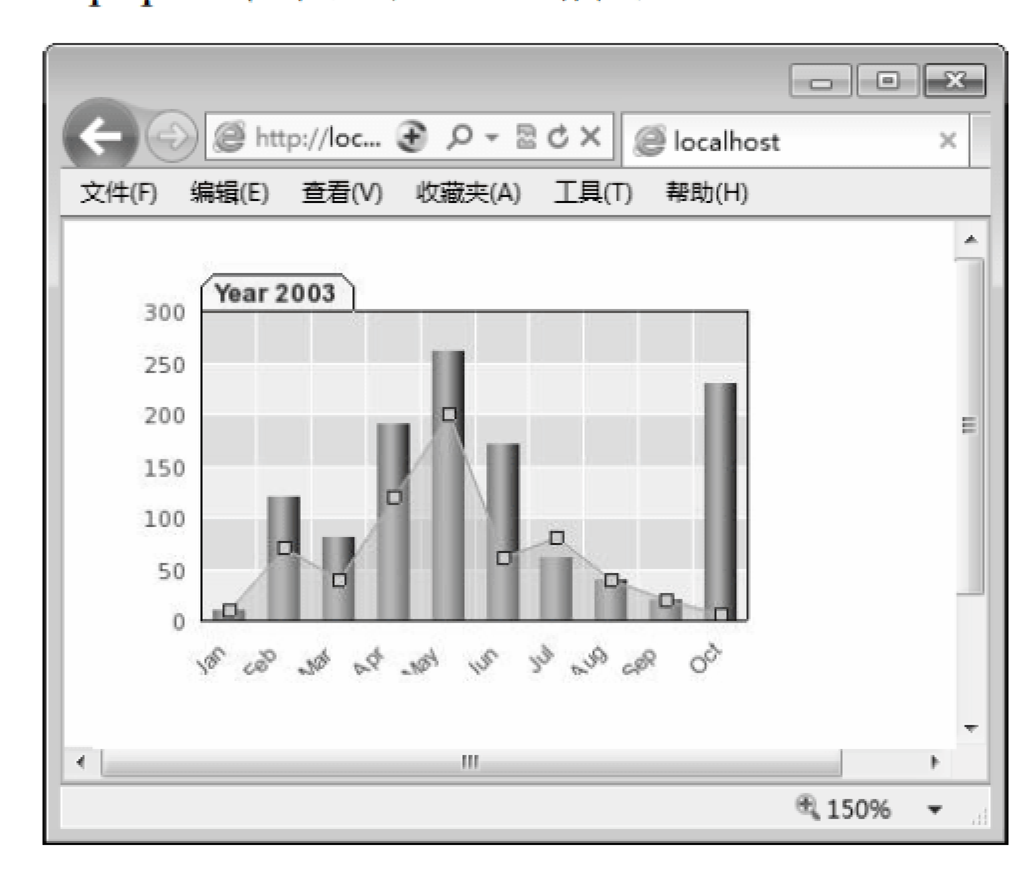


图 12-7　程序运行效果

【案例分析】：

（1） 其中 require_once ('jpgraph/src/jpgraph.php');、require_once ('jpgraph/src/jpgraph_bar.php');和 require_once ('jpgraph/src/jpgraph_line.php');语句加载了 jpgraph 基本类库 jpgraph.php、柱状图类库 jpgraph_bar.php 和折线图类库/jpgraph_line.php。

（2）$ydata= array(10,120,80,190,260,170,60,40,20,230);和$ydata2 = array(10,70,40,120,200,60,80,40,20,5);语句定义了柱状图和折线图在 y 轴上的数据坐标，这也是图形要表示的主要信息。

（3）$months = $gDateLocale->GetShortMonth();定义了月份使用短名表示。

（4）$graph = new Graph(300,200);语句创建图形$graph，高 300 像素，宽 200 像素。

（5）$graph->SetScale("textlin");语句确认刻度为自动生成的刻度形式。$graph-> SetMarginColor('white');语句确认图形边框颜色为白色。

（6）$graph->SetMargin(30,1,20,5);语句调整边框宽度。$graph->SetBox();语句在背景图上添加边框。$graph->SetFrame(false);语句取消整个图片的边框。

（7）$graph->tabtitle->Set('Year 2003');语句添加图片标题。$graph->tabtitle->SetFont(FF_ARIAL,FS_BOLD,10);语句设定标题样式。

（8）$graph->ygrid->SetFill(true,'#DDDDDD@0.5','#BBBBBB@0.5');语句设定 y 轴方向上的网格填充颜色和亮度。$graph->ygrid->SetLineStyle('dashed'); 语句设定 y 轴方向上的网格线的样式为虚线。$graph->ygrid->SetColor('gray'); 语句设定 y 轴方向上的网格线的颜色。$graph->xgrid->Show();语句、$graph->xgrid->SetLineStyle('dashed');语句和$graph->xgrid-> SetColor('gray');语句是对 x 轴方向网格的同理设定。

（9）$graph->xaxis->SetTickLabels($months);语句为对 x 轴的设定，它使用的是先前定义的$months 变量中的数据。$graph->xaxis->SetFont(FF_ARIAL,FS_NORMAL,8);语句设定样式。$graph->xaxis->SetLabelAngle(45);语句设定角度。

（10）$bplot = new BarPlot($ydata);语句采用先前的$ydata 数据生成柱状图。$bplot ->SetWidth(0.6);定义宽度。$bplot->SetFillGradient ($fcol,$tcol,GRAD_LEFT_REFLECTION); 填充柱状图，并且使用填充的渐变样式和两个渐变的颜色。$graph->Add($bplot);语句添加柱状图到图形中。

（11）$lplot = new LinePlot($ydata2);语句用$ydata2 数组生成折线图。$lplot->SetFillColor('skyblue@0.5');和$lplot->SetColor('navy@0.7');语句定义折线区域的颜色和透明度。

（12）$lplot->mark->SetType(MARK_SQUARE);定义折线图标记点的类型。$lplot->mark->SetColor('blue@0.5');定义颜色和透明度。$lplot->mark->SetSize(6);定义大小。$graph->Add($lplot);语句添加折线图到图形中。

（13）$graph->Stroke();语句表示把此图传递到浏览器显示。

12.3.4 制作圆形统计图

下面就通过圆形统计图实例的介绍来了解 Jpgraph 类的使用，具体步骤如下。

01 找到安装过的 jpgraph 类库文件夹，在其下的 src 文件夹下找到 Examples 文件夹。找到 balloonex1.php 文件，将其复制到 ch12 文件夹下，其代码如下。

```
<?php
require_once ('jpgraph/jpgraph.php');
```

```
require_once ('jpgraph/jpgraph_scatter.php');
$datax = array(1,2,3,4,5,6,7,8);
$datay = array(12,23,95,18,65,28,86,44);
function FCallback($aVal) {
    // This callback will adjust the fill color and size of
    // the datapoint according to the data value according to
    if( $aVal < 30 ) $c = "blue";
    elseif( $aVal < 70 ) $c = "green";
    else $c="red";
    return array(floor($aVal/3),"",$c);
}
$graph = new Graph(400,300,'auto');
$graph->SetScale("linlin");
$graph->img->SetMargin(40,100,40,40);
$graph->SetShadow();
$graph->title->Set("Example of ballon scatter plot");
$graph->yaxis->scale->SetGrace(50,10);
$graph->xaxis->SetPos('min');
$sp1 = new ScatterPlot($datay,$datax);
$sp1->mark->SetType(MARK_FILLEDCIRCLE);
$sp1->value->Show();
$sp1->value->SetFont(FF_FONT1,FS_BOLD);
$sp1->mark->SetCallback("FCallback");
$sp1->SetLegend('Year 2002');
$graph->Add($sp1);
$graph->Stroke();
?>
```

02 修改 require_once ('jpgraph/jpgraph.php');为 require_once ('jpgraph/src/jpgraph.php');。修改 require_once ('jpgraph/jpgraph_scatter.php');为 require_once ('jpgraph/src/jpgraph_scatter.php');。以载入本机 jpgraph 类库。

03 运行 balloonex1.php，结果如图 12-8 所示。

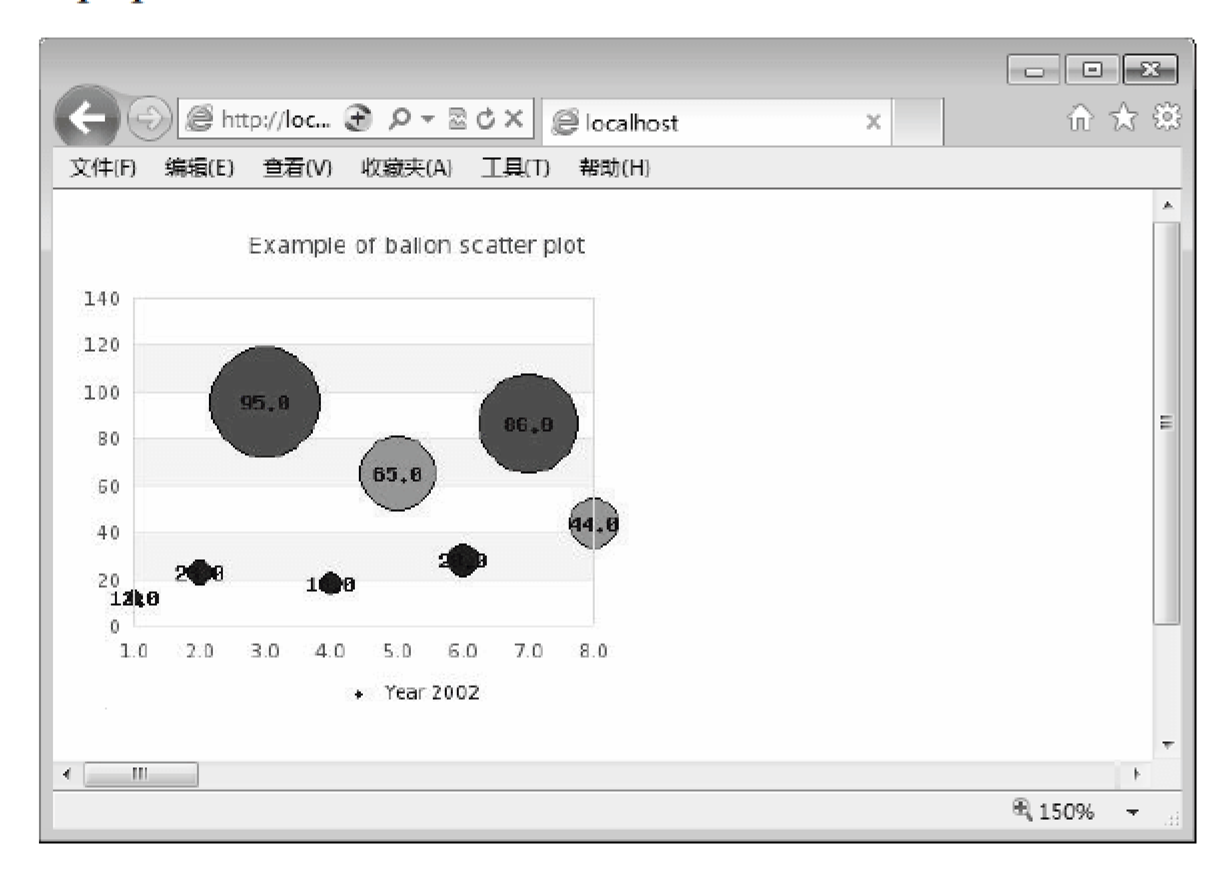


图 12-8　balloonex1.php 页面效果

【案例分析】：

（1）其中，require_once ('jpgraph/src/jpgraph.php');语句和 require_once ('jpgraph/src/jpgraph_scatter.php');语句加载了 jpgraph 基本类库 jpgraph.php 和圆形图类库 jpgraph_bar.php。

（2）$datax 和$datay 定义了两组要表现的数据。

（3）function FCallback($aVal){}函数定义了不同数值范围内的图形的颜色。

（4）$graph = new Graph(400,300,'auto');语句生成图形。$graph->SetScale("linlin");生成刻度。$graph->img->SetMargin(40,100,40,40); 设置图形边框。$graph->SetShadow(); 设置阴影。$graph->title->Set("Example of ballon scatter plot");设置标题。$graph->xaxis->SetPos('min');设置 x 轴为的位置为初始值。

（5）$sp1 = new ScatterPlot($datay,$datax); 生成数据表示图。$sp1->mark->SetType(MARK_FILLEDCIRCLE); 设置数据表示图的类型。$sp1->value->Show(); 展示数据表示图。$sp1->value->SetFont(FF_FONT1,FS_BOLD);设定展示图的字体。$sp1->SetLegend('Year 2002');设置标题。

（6）$graph->Add($sp1);语句添加数据展示图到整体图形中。

（7）$graph->Stroke();语句表示把此图传递到浏览器显示。

12.4 实战演练——制作 3D 饼形统计图

下面就通过 3D 饼形图实例的介绍，来了解 Jpgraph 类的使用方法和技巧，具体步骤如下。

01 找到安装过的 jpgraph 类库文件夹，在其下的 src 文件夹下找到 Examples 文件夹。找到 pie3dex3.php 文件，将其复制到 ch12 文件夹下，打开代码如下。

```
<?php
require_once ('jpgraph/ jpgraph.php');
require_once ('jpgraph/jpgraph_pie.php');
require_once ('jpgraph/jpgraph_pie3d.php');
$data = array(20,27,45,75,90);
$graph = new PieGraph(450,200);
$graph->SetShadow();
$graph->title->Set("Example 1 3D Pie plot");
$graph->title->SetFont(FF_VERDANA,FS_BOLD,18);
$graph->title->SetColor("darkblue");
$graph->legend->Pos(0.5,0.8);
$p1 = new PiePlot3d($data);
$p1->SetTheme("sand");
$p1->SetCenter(0.4);
$p1->SetAngle(30);
$p1->value->SetFont(FF_ARIAL,FS_NORMAL,12);
$p1->SetLegends(array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep"
,"Oct"));
$graph->Add($p1);
```

```
    $graph->Stroke();
    ?>
```

02 修改 require_once ('jpgraph/jpgraph.php');为 require_once ('jpgraph/src/jpgraph.php');。修改 require_once ('jpgraph/jpgraph_pie.php'); 为 require_once ('jpgraph/src/ jpgraph_pie.php ');。修改 require_once ('jpgraph/jpgraph_pie3d.php');为 require_once ('jpgraph/src/ jpgraph_pie3d.php ');。以载入本机 jpgraph 类库。

03 运行 pie3dex3.php，结果如图 12-9 所示。

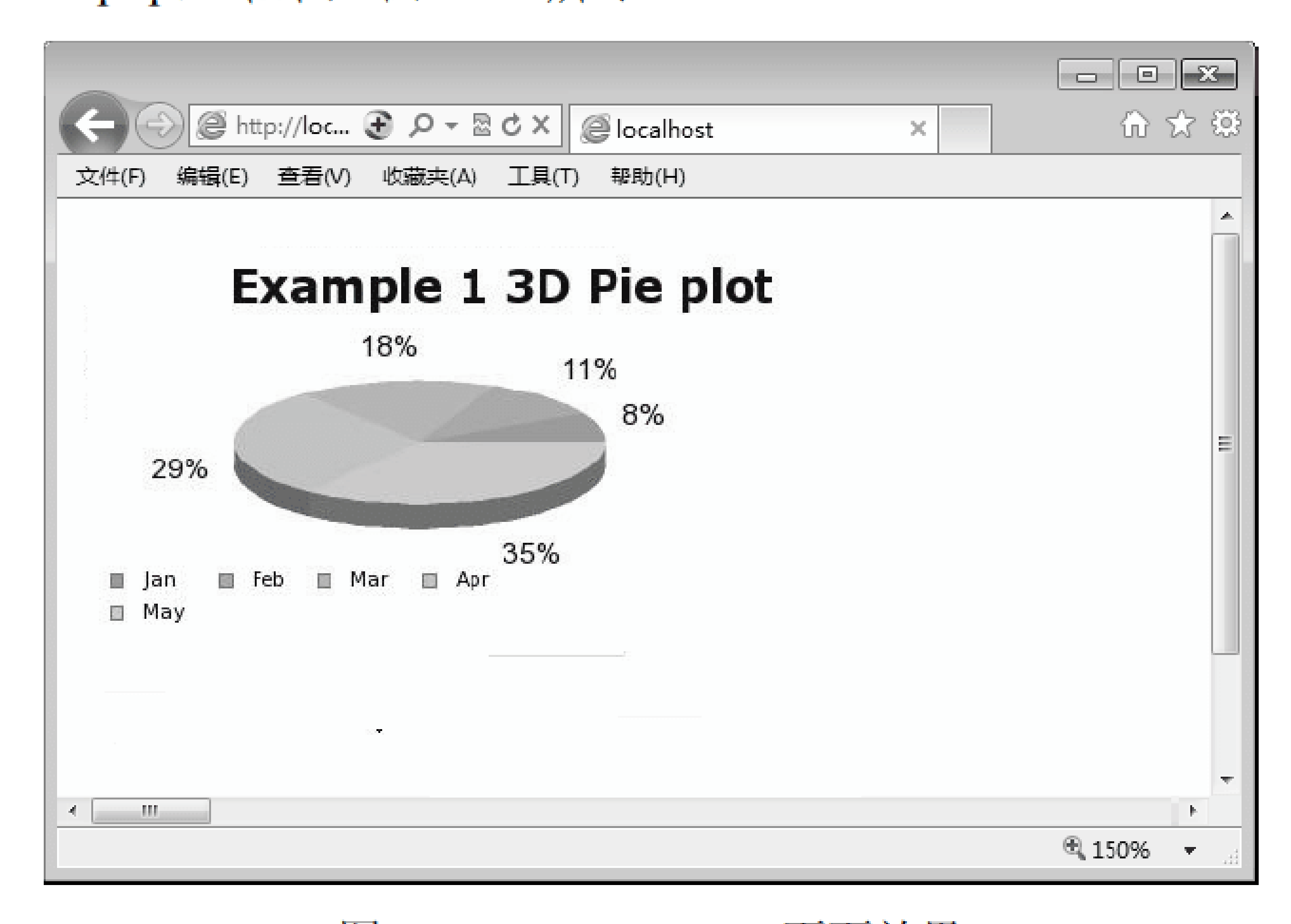


图 12-9 pie3dex3.php 页面效果

【案例分析】：

（1）其中 require_once ('jpgraph/src/jpgraph.php');语句、require_once ('jpgraph/jpgraph_pie.php');语句和 require_once ('jpgraph/jpgraph_pie3d.php');语句加载了 jpgraph 基本类库 jpgraph.php、饼形图类库 jpgraph_ pie.php 和 3d 饼形图类库 jpgraph_ pie3d.php。

（2）$data 定义了要表现的数据。

（3）$graph = new PieGraph(450,200);生成图形。$graph->SetShadow();设定阴影。

（4）$graph->title->Set("Example 1 3D Pie plot");设定标题。$graph->title->SetFont (FF_VERDANA, FS_BOLD,18); 设定字体和字体大小。$graph->title->SetColor("darkblue"); 设定颜色。$graph->legend->Pos(0.5,0.8);设定图例在整个图形中的位置。

（5）$p1 = new PiePlot3d($data);生成饼形图。$p1->SetTheme("sand");设置饼形图模板。$p1->SetCenter(0.4);设置饼形图的中心。$p1->SetAngle(30);设置饼形图角度。$p1->value->SetFont (FF_ARIAL,FS_NORMAL,12);设置字体。$p1->SetLegends(array("Jan",….,"Oct"));设置图例文字信息。

（6）$graph->Add($p1);向整个图形添加饼形图。$graph->Stroke();把此图传递到浏览器显示。

12.5 高手私房菜

技巧 1：不同格式的图片在使用上有何区别？

答：JPEG 格式是一个标准。JPEG 格式经常用来存储照片和拥有很多颜色的图片。它不强调压缩，强调的是对图片信息的保存。如果使用图形编辑软件缩小 jpeg 格式的图片，那么它原本所包含的一部分数据就会丢失。并且这种数据的丢失，通过肉眼是可以察觉到的。这种格式不适合包含简单图形颜色或文字的图片。

PNG 格式是指 Portable Network Graghics，这种图片格式是用来取代 gif 格式的。同样，图片使用 png 格式的大小要小于使用 gif 格式的大小。这种格式是一种低损失压缩的网络文件格式。这种格式的图片适合于包含文字、直线或色块的信息。PNG 支持透明、伽马校正等。但是 png 不像 gif 一样支持动画功能，并且 IE 6 不支持 png 的透明功能。低损压缩意味着压缩比不高，所以它不适合用于照片这类的图片，否则文件将太大。

GIF 是指 Graphics Interchange Format，它也是一种低损压缩的格式，适合用于包含文字、直线或色块的信息的图片。它使用的是 24 位 RGB 色彩中的 256 色。由于色彩有限所以也不适合用于照片一类的大图片。对于其适合的图片，它具有不丧失图片质量却能大幅压缩的图片大小的优势。另外，它支持动画。

技巧 2：如何选择自己想要的 RGB 颜色呢？

可以使用 Photoshop 里面的颜色选取工具获取想要的颜色。如果使用的是 Linux 系统，可以使用开源的工具 GIMP 中的颜色选取工具。

12.6 经典习题

（1）制作一个包含读取指定文件的例子。
（2）制作一个包含写入数据到文件的例子。
（3）制作一个读取文件目录的例子。
（4）制作一个图片文件上传的例子。
（5）制作一个记录访客计算器的例子。

第 13 章　Cookie 和会话管理

由于 HTTP Web 协议是无状态协议，对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。客户端与服务器进行动态交互的 Web 应用程序出现之后，HTTP 无状态的特性严重阻碍了这些应用程序的实现，毕竟交互是需要承前启后的，简单的购物车程序也要知道用户到底在之前选择了什么商品。于是，两种用于保持 HTTP 连接状态的技术就应运而生了，一个是 Cookie，而另一个则是 SESSION。其中 Cookie 将数据存储在客户端，并显示永久的数据存储。Session 将数据存储在服务器端，保证数据在程序的单次访问中持续有效。本章主要讲述 Cookie 和 Session 的使用方法和应用技巧。

本章学习目标

- 掌握 Cookie 的基本操作方法
- 熟悉 Session 的基本概念
- 熟悉 Cookie 和 Session 的区别和联系
- 掌握会话管理的基本操作

13.1　Cookie 基本操作

下面介绍 Cookie 的含义和基本用法。

13.1.1　什么是 Cookie

Cookie 常用于识别用户。Cookie 是服务器留在用户计算机中的小文件。每当相同的计算机通过浏览器请求页面时，它同时会发送 Cookie。

Cookie 的工作原理是：当一个客户端浏览器连接到一个 URL，它会首先扫描本地存储的 Cookie，如果发现其中有和此 URL 相关联的 Cookie，将会把它们返回给服务器端。

Cookie 通常应用于以下几个方面。

（1）在页面之间传递变量。因为浏览器不会保存当前页面上的任何变量信息，如果页面被关闭，则页面上的所有变量信息也会消失。通过 Cookie，可以把变量值在 Cookie 中保存下来，然后另外的页面可以重新读取这个值。

（2）记录访客的一些信息。利用 Cookie 可以记录客户曾经输入的信息或者记录访问网页的次数。

（3）通过把所查看的页面保存在 Cookie 临时文件夹中，可以提高以后的浏览速度。

用户可以通过 header 以如下格式在客户端生成 Cookie：

```
Set-cookie:NAME = VALUE;[expires=DATE;][path=PATH;][domain=DOMAIN_NAME;]
[secure]
```

NAME 为 cookie 的名称，VALUE 为 cookie 的值，expires=DATE 为到期日，path=PATH;

domain=DOMAIN_NAME;为与某个地址相对应的路径和域名，secure 表示 cookie 不能通过单一的 HTTP 连接传递。

13.1.2　创建 Cookie

通过 PHP，用户能够创建 Cookie。创建 Cookie 使用 setcookie()函数，它的语法格式如下：

```
setcookie（名称，cookie 值，到期日，路径，域名，secure）
```

其中的参数与 Set-cookie 中的参数意义相同。



setcookie()函数必须位于<html>标签之前。

在下面的例子中，将创建名为 user 的 cookie，为它赋值为“Cookie 保存的值”，并且规定此 cookie 在一小时后过期。

【例 13.1】（实例文件：ch13\13.1.php）

```
<?php
setcookie("user", " Cookie 保存的值", time()+3600);
?>
<html>
<body>
</body>
</html>
```

运行上述程序，会在 Cookies 文件夹下自动生成一个 Cookie 文件，有效期为一个小时，在 Cookie 失效后，Cookies 文件自动被删除。



如果用户没有设置 Cookie 的到期时间，则在关闭浏览器时会自动删除 Cookie 数据。

13.1.3　读取 Cookie

那么如何取回 Cookie 的值呢？在 PHP 中，使用$_COOKIE 变量取回 Cookie 的值。下面通过实例讲解如何取回上面创建的名为 user 的 Cookie 的值，并把它显示在页面上。

【例 13.2】（实例文件：ch13\13.2.php）

```
<?php
// 输出一个 cookie
echo $_COOKIE["user"];
// 显示所有的 cookie
```

```
print_r($_COOKIE);
?>
```

程序运行效果如图 13-1 所示。

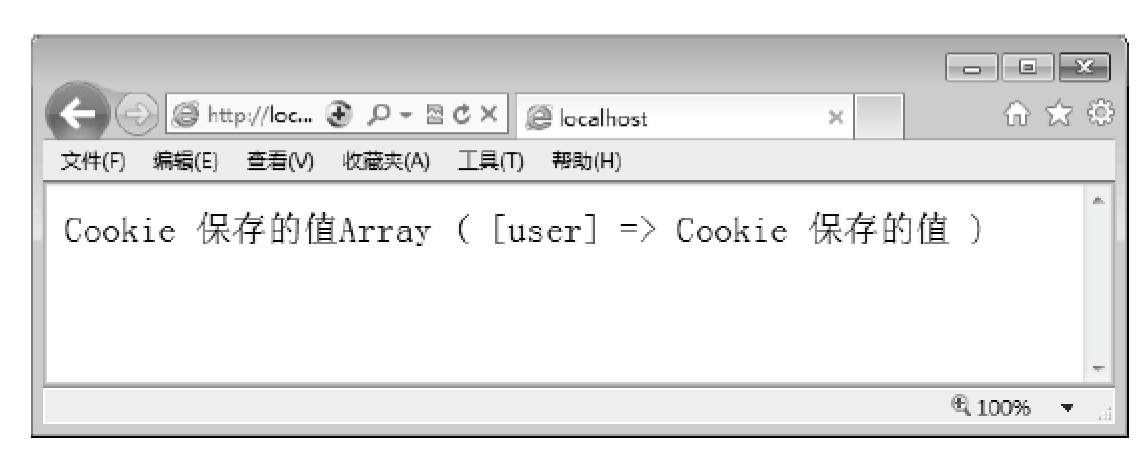


图 13-1　程序运行效果

用户可以通过 isset()函数来确认是否已设置 cookie。下面通过实例来讲解。

【例 13.3】（实例文件：ch13\13.3.php）

```
<html>
<body>
<?php
  if (isset($_COOKIE["user"]))                              //如果 Cookie 文件存在
    echo "Welcome " . $_COOKIE["user"] . "!<br />";
  else                                                      //如果 Cookie 文件不存在
    echo "Welcome guest!<br />";
?>
</body>
</html>
```

程序运行效果如图 13-2 所示。

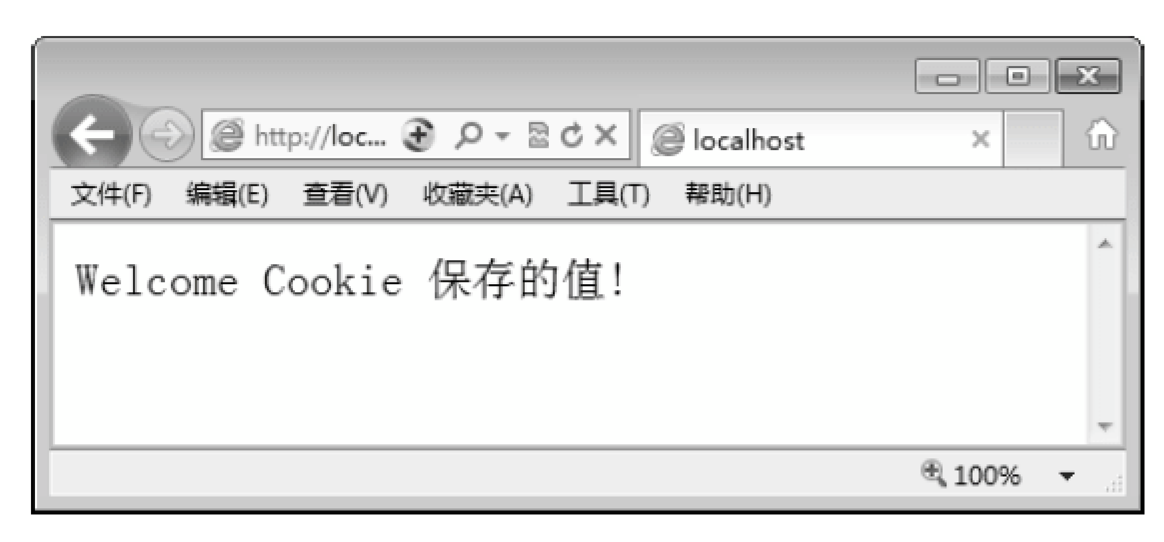


图 13-2　程序运行效果

13.1.4　删除 Cookie

常见的删除 Cookie 的方法有两种，包括在浏览器中手动删除和使用函数删除。

1. 在浏览器中手动删除

由于 Cookie 自动生成的文本会存在于 IE 浏览器的 Cookies 临时文件夹中，在浏览器中删除 Cookie 文件是比较快捷的方法。具体的操作步骤如下。

01 在浏览器页面中选择【工具】选项，在弹出的下拉菜单中选择【Internet 选项】菜单命令，如图 13-3 所示。



图 13-3　选择【Internet 选项】菜单命令

02 打开【Internet 选项】对话框，在【常规】选项卡中单击【删除】按钮，如图 13-4 所示。

03 打开【删除浏览的历史记录】对话框，选中【Cookie】复选框，单击【删除】按钮即可，如图 13-5 所示。返回到【Internet 选项】对话框，单击【确定】按钮即可完成删除 Cookie 的操作。

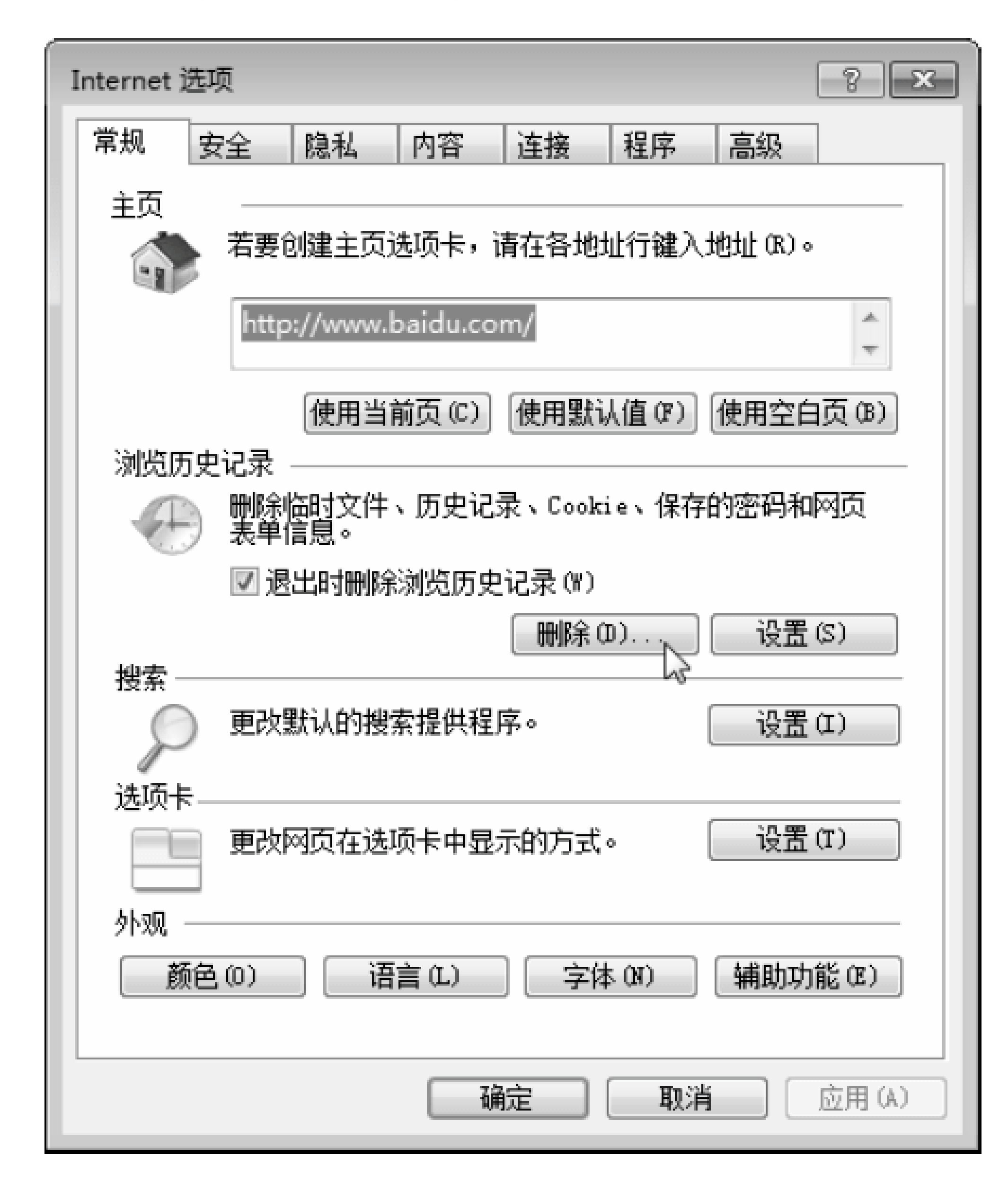


图 13-4　【Internet 选项】对话框

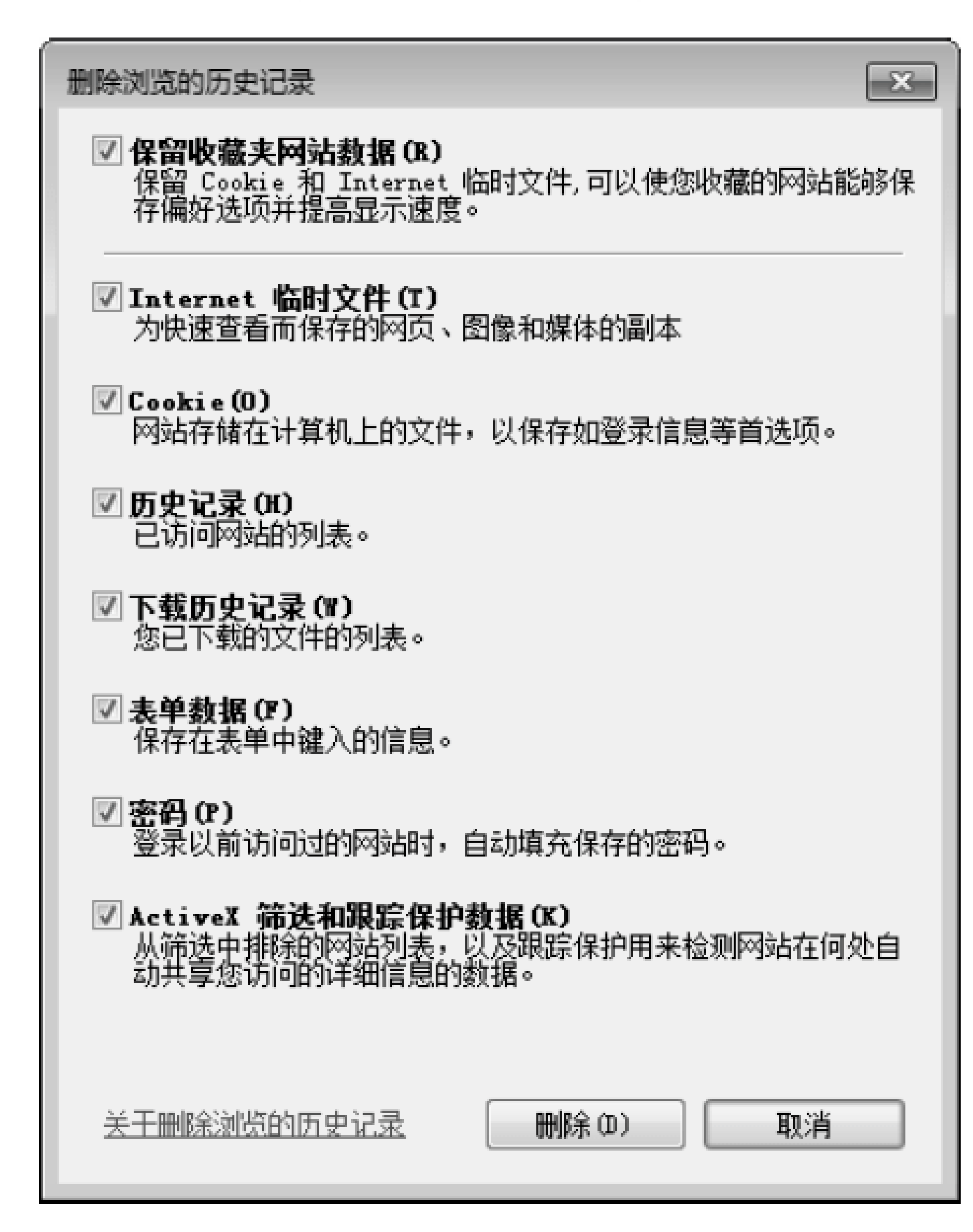


图 13-5　【删除浏览的历史记录】对话框

2. 使用函数删除

删除 Cookie 仍然使用 setcookie()函数。当删除 Cookie 时，将第二个参数设置为空，第三个参数的过期时间设置为小于系统的当前时间即可。

【例 13.4】（实例文件：ch13\13.4.php）

```
<?php
  //将 Cookie 的过期时间设置为比当前时间减少 10 秒
  setcookie("user", "", time()-10);
?>
```

在上面的代码中，time()函数返回的是当前的系统时间，把过期时间减少 10 秒，这样过期时间就会变成过去的时间，从而删除 Cookie。如果将过期时间设置为 0，也可以直接删除 Cookie。

13.2 认识 Session

下面介绍 Session 的一些基本概念和使用方法。

13.2.1 什么是 Session

由于 HTTP 是无状态协议，也即是说 HTTP 的工作过程是请求与回应的简单过程，所以 HTTP 没有一个内置的方法来存储这个过程中各方的状态。比如，当同一个用户向服务器发出两个不同的请求时，虽然服务器端都会给以相应的回应，但是它并没有办法知道这两个动作是由同一个用户发出的。

由此，会话（Session）管理应运而生。通过使用一个会话，程序可以跟踪用户的身份和行为，并且根据这些状态数据，给用户以相应的回应。

13.2.2 Session 的基本功能

在 PHP 中，每一个 Session 都有一个 ID。这个 Session ID 是一个由 PHP 随机生成的加密数字。这个 Session ID 通过 cookie 存储在客户端浏览器中，或者直接通过 URL 传递到客户端，如果在某个 URL 后面看到一长串加密的数字，这很有可能就是 Session ID 了。

Session ID 就像一把钥匙用来注册到 session 变量中。而这些 session 变量是存储在服务器端的。Session ID 是客户端唯一存在的会话数据。

使用 Session ID 打开服务器端相对应的 session 变量，跟用户相关的会话数据便一目了然。默认情况下，在服务器端的 session 变量数据是以文件的形式加以存储的，但是会话变量数据也经常通过数据库进行保存。

13.2.3 Cookie 与 Session

在浏览器中，有些用户出于安全性的考虑，关闭了其浏览器的 Cookie 功能。Cookie 将不能正常工作。

使用 Session 可以不需要手动设置 Cookie，PHP Session 可以自动处理。可以使用会话管理，及 PHP 中的 session_get_cookie_params()函数来访问 Cookie 的内容。这个函数将返回一个数组，包括 Cookie 的生存周期、路径、域名、secure 等。它的格式为：

```
session_get_cookie_params（生存周期、路径、域名、secure）
```

13.2.4　在 Cookie 或 URL 中存储 Session ID

PHP 默认情况下会使用 Cookie 来存储 sessio ID。但是如果客户端浏览器不能正常工作，就需要用 URL 方式传递 session ID。如果在 php.ini 中的 session.use_trans_sid 设置为启用的状态，就可以自动通过 URL 来传递 session ID。

不过通过 URL 传递 session ID 会产生一些安全问题。如果这个连接被其他用户复制并使用，有可能造成用户判断的错误。其他用户可能使用 session ID 访问目标用户的数据。

或者可以通过程序把 session ID 存储到常量 SID 中，然后通过一个连接传递。

13.3　会话管理

一个完整的会话包括创建会话、注册会话变量、使用会话变量和删除会话变量。下面介绍有关会话管理的基本操作。

13.3.1　创建会话

常见的创建会话的方法有 3 种，包括 PHP 自动创建、使用 session_start()函数创建和使用 session_register()函数创建。

1. PHP 自动创建

用户可以在 php.ini 中设定 session.auto_start 为启用。但是使用这种方法的同时，不能把 session 变量对象化。应定义此对象的类，必须在创建会话之前加载，然后新创建的会话才能加载此对象。

2. 使用 session_start()函数

这个函数首先检查当前是否已经存在一个会话，如果不存在，它将创建一个全新的会话，并且这个会话可以访问超全局变量$_SESSION 数组。如果已经有一个存在的会话，函数会直接使用这个会话，加载已经注册过的会话变量，然后使用。

session_start()函数的语法格式如下：

```
bool session_start(void);
```



提　示

session_start()函数必须位于<html>标签之前。

【例 13.5】（实例文件：ch13\13.5.php）

```
<?php
  session_start();
 ?>
<html>
<body>
```

```
</body>
</html>
```

上面的代码会向服务器注册用户的会话，以便可以开始保存用户信息，同时会为用户会话分配一个 UID。

3. 使用 session_register()函数

在使用 session_register()函数之前，需要在 php.ini 文件中将 register_globals 设置为 on，然后需要重启服务器。session_register()函数通过为会话登录一个变量来隐含地启动会话。

13.3.2 注册会话变量

会话变量被启动后，全部保存在数组$_SESSION 中。用户可以通过对$_SESSION 数组赋值来注册会话变量。

例如，启动会话，创建一个 Session 变量并赋予 xiaoli 的值，代码如下：

```
<?php
  session_start();                      //启动 Session
  $_SESSION['name']='xiaoli';           //声明一个名为 name 的变量，并赋值 xiaoli
 ?>
```

这个会话变量值会在此会话结束或被注销后失效。或者还会根据 php.ini 中的 session.gc_maxlifetime（当前系统设置的 1440 秒，也就是 24 小时）设置会话最大生命周期数，过期而失效。

13.3.3 使用会话变量

使用会话变量，首先要判断会话变量是否存在一个会话 ID，如果不存在，则需要创建一个，并且能够通过$_SESSION 变量进行访问。如果已经存在，则将这个已经注册的会话变量载入以供用户使用。

在访问$_SESSION 数组时，先要使用 isset()或 empty()来确定$_SESSION 中会话变量是否为空。

例如下面的代码所示：

```
<?php
  If (! empty ($_SESSION['session_name'])) ;      //判断会话变量是否为空
  $ssvalue=$_SESSION['session_name'];             //声明一个名为 name 的变量，并赋
值 xiaoli
 ?>
```

下面通过实例讲解存储和取回$_SESSION 变量的方法。

【例 13.6】（实例文件：ch13\13.6.php）

```
<?php
```

```
session_start();
// 存储会话变量的值
$_SESSION['views']=1;
?>
<html>
<body>
<?php
//读取会话变量的值
echo "浏览量=". $_SESSION['views'];
?>
</body>
</html>
```

程序运行效果如图 13-6 所示。



图 13-6 程序运行效果

13.3.4 注销和销毁会话变量

注销会话变量使用 unset()函数即可，如 unset（$_SESSION['name']）。（不再需要使用 php4 中的 session_unrgister()或 session_unset()）。unset()函数用于释放指定的 session 变量，代码如下：

```
<?php
unset($_SESSION['views']);
?>
```

如果要注销所有会话变量，只需要向$_SESSION 赋值一个空数组就可以了，如$_SESSION = array()。注销完成后，使用 session_destory()销毁会话即可，其实就是清除相应的 sessionID。代码如下：

```
<?php
  session_destroy();
?>
```

13.4 实战演练——会话管理的综合应用

下面通过一个综合案例，讲述会话管理的综合应用。

01 在网站根目录下建立一个文件夹，名为 session。

02 在 session 文件夹下建立 opensession.php，输入以下代码并保存。

```
<?php
  session_start();
  $_SESSION['name'] = "王小明";
   echo "会话变量为:".$_SESSION['name'];
?>
<a href='usesession.php'>下一页</a>
```

03 在 session 文件夹下建立 usesession.php，输入以下代码并保存。

```
<?php
  session_start();
  echo "会话变量为:".$_SESSION['name']."<br />";
  echo $_SESSION['name'].",你好。";
?>
<a href='closesession.php'>下一页</a>
```

04 在 session 文件夹下建立 closesession.php，输入以下代码并保存。

```
<?php
  session_start();
  unset($_SESSION['name']);
  if (isset($_SESSION['name'])){
     echo "会话变量为:".$_SESSION['name'];
  }else{
     echo "会话变量已注销。";
  }
  session_destroy();
?>
```

05 运行 opensession.php 文件，结果如图 13-7 所示。



图 13-7 程序运行结果

06 单击页面中的“下一页”链接，运行结果如图 13-8 所示。

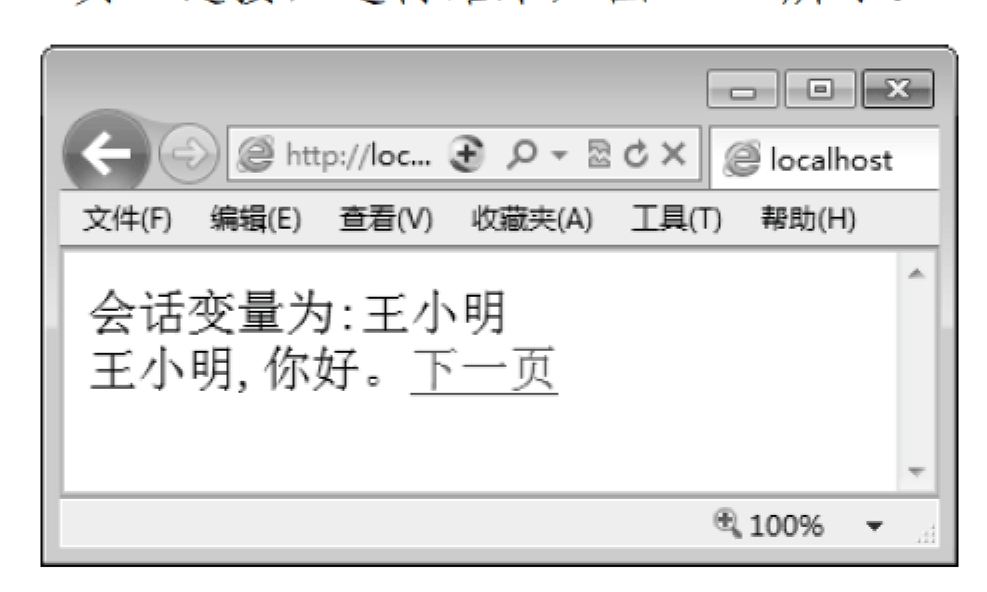


图 13-8 程序运行结果

07 单击页面中的“下一页”链接，运行结果如图 13-9 所示。



图 13-9 程序运行结果

13.5 高手私房菜

技巧 1：如果浏览器不支持 Cookie，怎么办？

如果应用程序涉及不支持 Cookie 的浏览器，不得不采取其他方法在应用程序中从一张页面向另一张页面传递信息。一种方式是从表单传递数据。

下面的表单在用户单击提交按钮时向 welcome.php 提交了用户输入：

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

要取回 welcome.php 中的值，可以设置如下代码：

```
<html>
<body>
```

```
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

技巧 2：Cookie 的生命周期是多久？

如果 Cookie 不设定失效时间，则表示它的生命周期为未关闭浏览器前的时间段，一旦浏览器关闭，Cookie 则会自动消失。

如果设定了过期时间，那么浏览器会把 Cookie 保存到硬盘中，在超过有效期前，用户打开 IE 浏览器会依然有效。

由于浏览器最多存储 300 个 Cookie 文件，每个 Cookie 文件最大支持 4KB，所以一旦超过容量的限制，则浏览器会自动随机地删除 Cookies。

13.6 经典习题

（1）制作一个创建、读取和删除 Cookie 的例子。
（2）在浏览器中手动删除 Cookie。
（3）制作一个通过 session_start()函数创建会话的例子。
（4）制作一包含注销和销毁会话变量的例子
（5）制作一个综合管理会话的例子。

第 14 章　MySQL 数据库基础

MySQL 是一个小型关系数据库管理系统，与其他大型数据库管理系统如 Oracle、DB2、SQL Server 等相比，MySQL 规模小、功能有限，但是它体积小、速度快、成本低，且它提供的功能对稍微复杂的应用来说已经够用，这些特性使得 MySQL 成为世界上最受欢迎的开放源代码数据库。MySQL 支持在多种平台下工作。在 Windows 平台下可以使用二进制的安装软件包或免安装版的软件包进行安装，二进制的安装包提供了图形化的安装向导过程，免安装版直接解压缩即可使用。本书以目前最新的版本 MySQL 5.7 为例进行讲解，主要讲述 MySQL 服务器的一些常见操作。

本章学习目标

- 熟悉什么是 MySQL
- 掌握启动和登录 MySQL 的方法
- MySQL 常用图形管理工具

14.1　什么是 MySQL

本节将介绍 MySQL 的基本知识。

14.1.1　客户机-服务器软件

主从式架构（Client-Server Model）或客户端/服务器（Client/Server）结构简称 C/S 结构，是一种网络架构，通常在该网络架构下软件分为客户端（Client）和服务器（Server）。

服务器是整个应用系统资源的存储与管理中心，多个客户端则各自处理相应的功能，共同实现完整的应用。在客户端/服务器结构中，客户端用户的请求被传送到数据库服务器，数据库服务器进行处理后，将结果返回给用户，从而减少了网络数据传输量。

用户使用应用程序时，首先启动客户端通过有关命令告知服务器进行连接以完成各种操作，而服务器则按照此请求提供相应的服务。每一个客户端软件的实例都可以向一个服务器或应用程序服务器发出请求。

这种系统的特点就是，客户端和服务器程序不在同一台计算机上运行，这些客户端和服务器程序通常归属于不同的计算机。

主从式架构通过不同的途径应用于很多不同类型的应用程序，比如，现在人们最熟悉的在因特网上使用的网页，当顾客想要在当当网站上买书的时候，电脑和网页浏览器就被当作一个客户端，同时，组成当当网的电脑、数据库和应用程序就被当作服务器。当顾客的网页浏览器向当当网请求搜寻数据库相关的图书时，当当网服务器从当当网的数据库中找出所有该类型的图书信息，结合成一个网页，再发送回顾客的浏览器。服务器端一般使用高性能的计算机，并配合使用不同类型的数据库，比如 Oracle、Sybase 或者是 MySQL 等；客户端需要安装专门的软件，比如浏览器。

14.1.2 MySQL 版本

针对不同用户，MySQL 分为两个不同的版本。

- MySQL Community Server（社区版）：该版本完全免费，但是官方不提供技术支持。
- MySQL Enterprise Server（企业版）：它能够以很高的性价比为企业提供数据仓库应用，支持 ACID 事物处理，提供完整的提交、回滚、崩溃恢复和行级锁定功能。但是该版本需付费使用，官方提供电话技术支持。


提 示

MySQL Cluster 主要用于架设集群服务器，需要在社区版或企业版基础上使用。

MySQL 的命名机制由 3 个数字和 1 个后缀组成，例如 mysql-5.7.10。

- 第 1 个数字（5）是主版本号，描述了文件格式，所有版本 5 的发行版都有相同的文件格式。
- 第 2 个数字（7）是发行级别，主版本号和发行级别组合在一起便构成了发行序列号。
- 第 3 个数字（10）是在此发行系列的版本号，随每次新分发版本递增。通常选择已经发行的最新版本。

在 MySQL 开发过程中，同时存在多个发布系列，每个发布处在成熟度的不同阶段。

- MySQL 5.7 是最新开发的稳定（GA）发布系列，是将执行新功能的系列，目前已经可以正常使用。
- MySQL 5.6 是比较稳定（GA）发布系列。只针对漏洞修复重新发布，没有增加会影响稳定性的新功能。
- MySQL 5.1 是前一稳定（产品质量）发布系列。只针对严重漏洞修复和安全修复重新发布，没有增加会影响该系列的重要功能。


提 示

对于 MySQL 4.1、4.0 和 3.23 等低于 5.0 的老版本，官方将不再提供支持。而所有发布的 MySQL（Current Generally Available Release）版本已经经过严格标准的测试，可以保证其安全可靠性。针对不同的操作系统，读者可以在 MySQL 官方下载页面（http://dev.mysql.com/downloads/）下载相应的安装文件。

14.1.3 MySQL 的优势

MySQL 的主要优势如下。

- 速度：运行速度快。
- 价格：MySQL 对多数个人用户来说是免费的。
- 容易使用：与其他大型数据库的设置和管理相比，其复杂程度较低，易于学习。
- 可移植性：能够工作在众多不同的系统平台上，例如 Windows、Linux、Unix、Mac OS 等。
- 丰富的接口：提供了用于 C、C++、Eiffel、Java、Perl、PHP、Python、Ruby 和 Tcl 等语言的 API。

- 支持查询语言：MySQL 可以利用标准 SQL 语法和支持 ODBC（开放式数据库连接）的应用程序。
- 安全性和连接性：十分灵活、安全的权限和密码系统，允许基于主机的验证。连接到服务器时，所有的密码传输均采用加密形式，从而保证了密码的安全。由于 MySQL 是网络化的，因此可以在因特网上的任何地方访问，提高数据共享的效率。

14.2 启动服务并登录 MySQL 数据库

用户可以下载 MySQL 并安装，安装完毕之后，需要启动服务器进程，否则客户端无法连接数据库，客户端通过命令行工具登录数据库。本节将介绍如何启动 MySQL 服务器和登录 MySQL 的方法。

14.2.1 启动 MySQL 服务

在前面的配置过程中，已经将 MySQL 安装为 Windows 服务，当 Windows 启动、停止时，MySQL 也自动启动、停止。不过，用户还可以使用图形服务工具来控制 MySQL 服务器或从命令行使用 NET 命令。

可以通过 Windows 的服务管理器查看，具体的操作步骤如下。

01 单击【开始】菜单，在弹出的菜单中选择【运行】命令，打开【运行】对话框，如图 14-1 所示。

02 在【打开】文本框中输入 services.msc，单击【确定】按钮，打开 Windows 的【服务管理器】，在其中可以看到服务名为 MySQL 的服务项，其右边的状态为“已启动”，表明该服务已经启动，如图 14-2 所示。

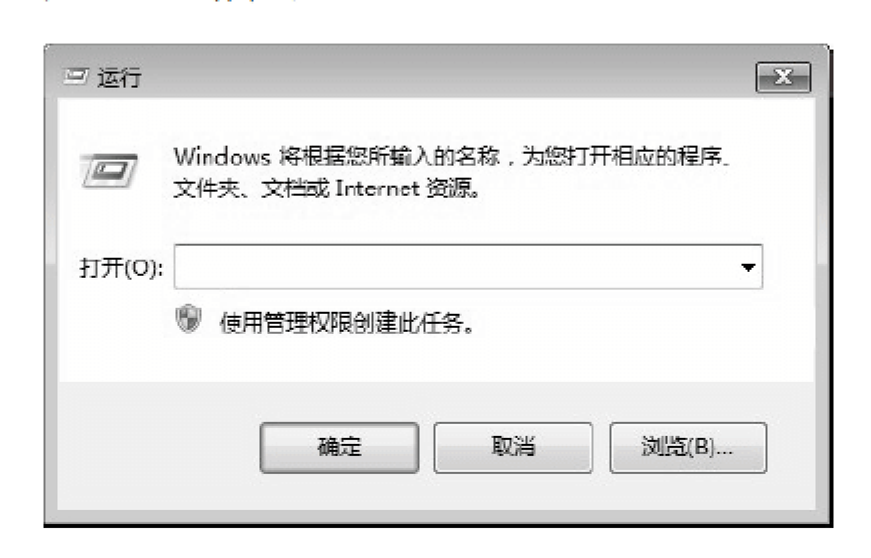


图 14-1 【运行】对话框



图 14-2 服务管理器窗口

由于设置了 MySQL 为自动启动，在这里可以看到，服务已经启动，而且启动类型为自动。如果没有“已启动”字样，说明 MySQL 服务未启动。启动方法为：单击【开始】菜单，选择【运行】命令，在【运行】对话框中输入 cmd，回车后弹出 Windows 7 命令提示符界面。然后输入“net start mysql”，按回车键，就能启动 MySQL 服务了，停止 MySQL 服务的命令为“net stop mysql”，如图 14-3 所示。



提 示

输入的 MySQL 是服务的名字。如果读者的 MySQL 服务的名字是 DB 或其他名字，应该输入“net start DB”或其他名称。

也可以直接双击 MySQL 服务，打开 MySQL 属性对话框，在其中通过单击【启动】或【停止】按钮来更改服务状态，如图 14-4 所示。



图 14-3　在命令行中启动和停止 MySQL　　图 14-4　MySQL 服务属性对话框

14.2.2　登录 MySQL 数据库

当 MySQL 服务启动完成后，便可以通过客户端来登录 MySQL 数据库了。在 Windows 操作系统下，可以通过两种方式登录 MySQL 数据库。

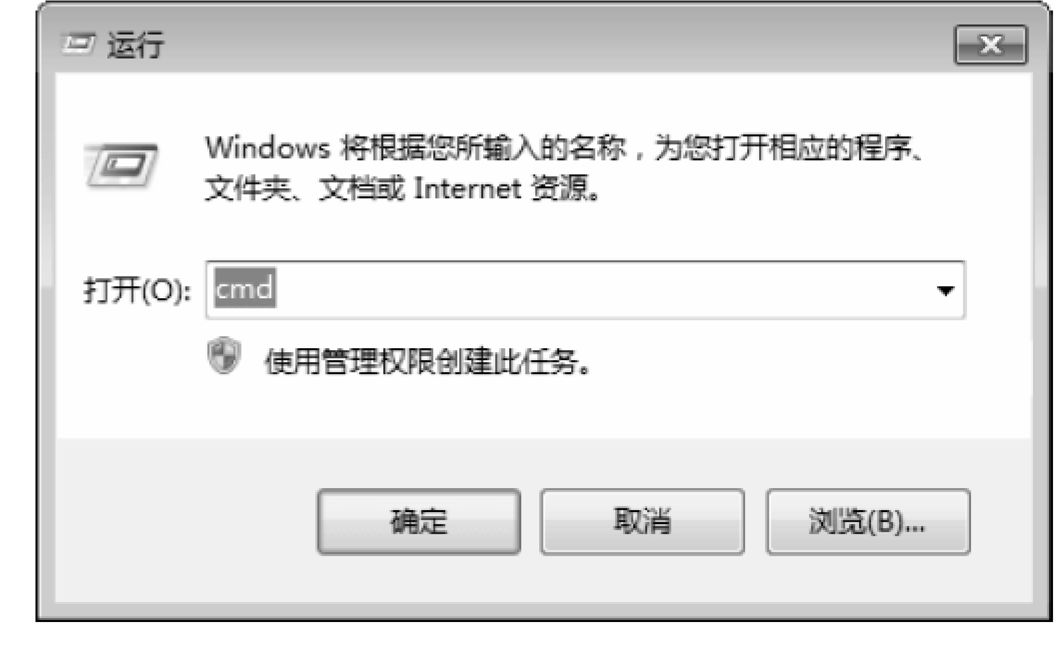
1. 以 Windows 命令行方式登录

具体的操作步骤如下。

01 单击【开始】按钮，在弹出的菜单中选择【运行】菜单命令，打开【运行】对话框，在其中输入命令 cmd，如图 14-5 所示。

02 单击【确定】按钮，打开 DOS 窗口，输入以下命令并按 Enter 键确认，如图 14-6 所示。

```
cd C:\Program Files\MySQL\MySQL Server 5.7\bin\
```



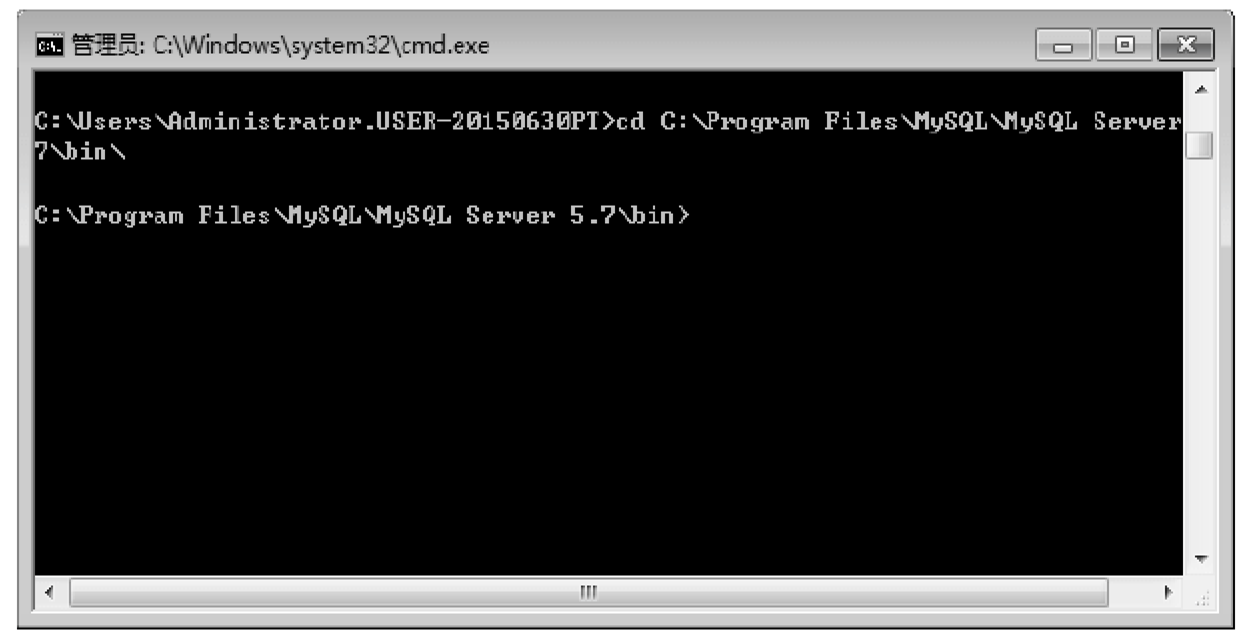


图 14-5　【运行】对话框　　图 14-6　DOS 窗口

03 在 DOS 窗口中可以通过登录命令连接到 MySQL 数据库，连接 MySQL 的命令格式为：

```
mysql -h hostname -u username -p
```

其中 MySQL 为登录命令，–h 后面的参数是服务器的主机地址，在这里客户端和服务器位于

同一台机器上，所以输入 localhost 或者 IP 地址 127.0.0.1，-u 后面跟登录数据库的用户名称，在这里为 root，-p 后面是用户登录密码。

接下来，输入如下命令：

```
mysql  -h localhost -u root -p
```

按回车键，系统会提示输入密码（Enter password），这里输入在前面配置向导中自己设置的密码，验证正确后，即可登录到 MySQL 数据库，如图 14-7 所示。

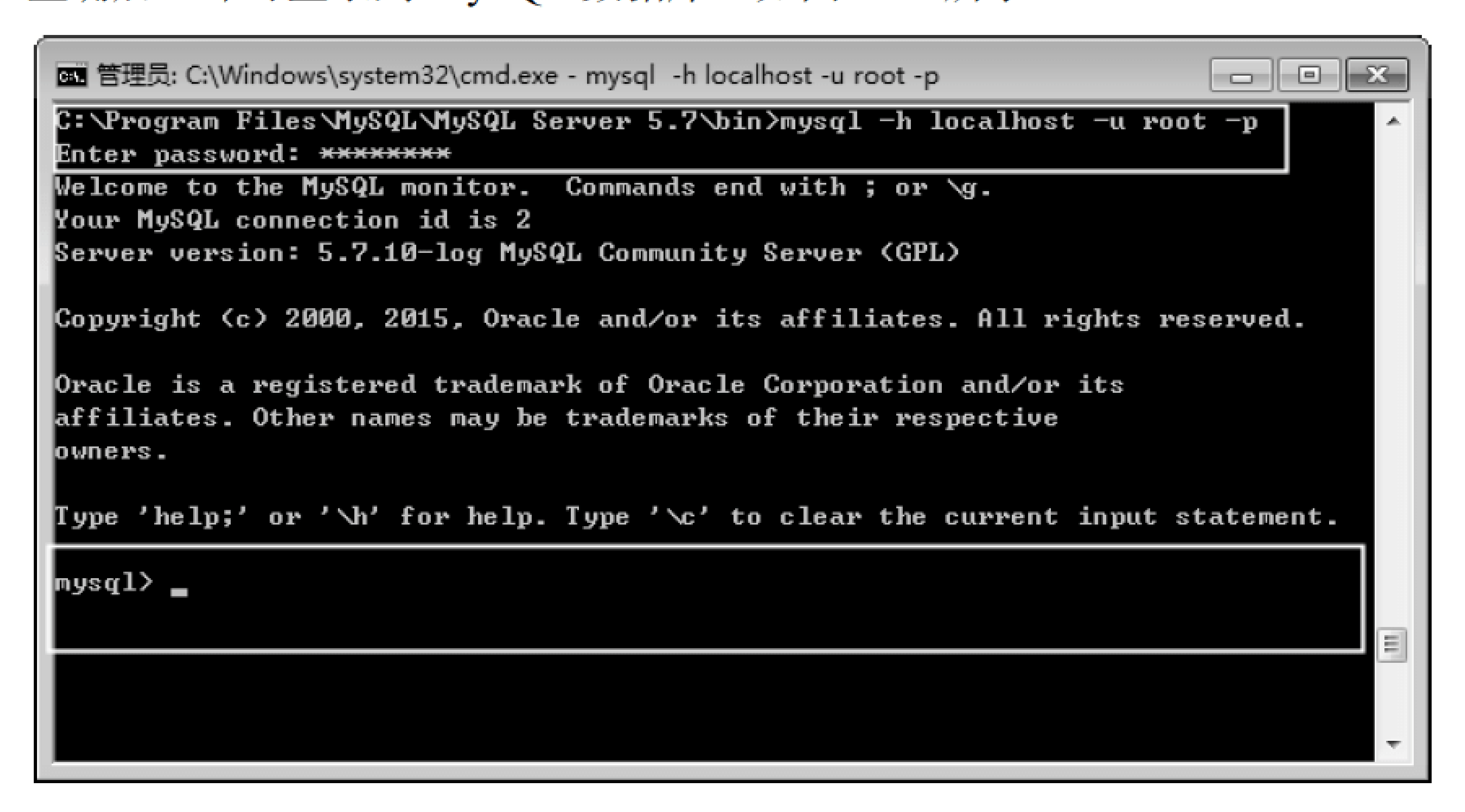


图 14-7　Windows 命令行登录窗口



提 示

当窗口中出现如图 14-7 所示的说明信息，且命令提示符变为“MySQL>”时，表明已经成功登录 MySQL 服务器，可以开始对数据库进行操作。

2. 使用 MySQL Command Line Client 登录

依次选择【开始】|【所有程序】|【MySQL】|【MySQL Server 5.7】|【MySQL 5.7 Command Line Client】菜单命令，进入密码输入窗口，如图 14-8 所示。

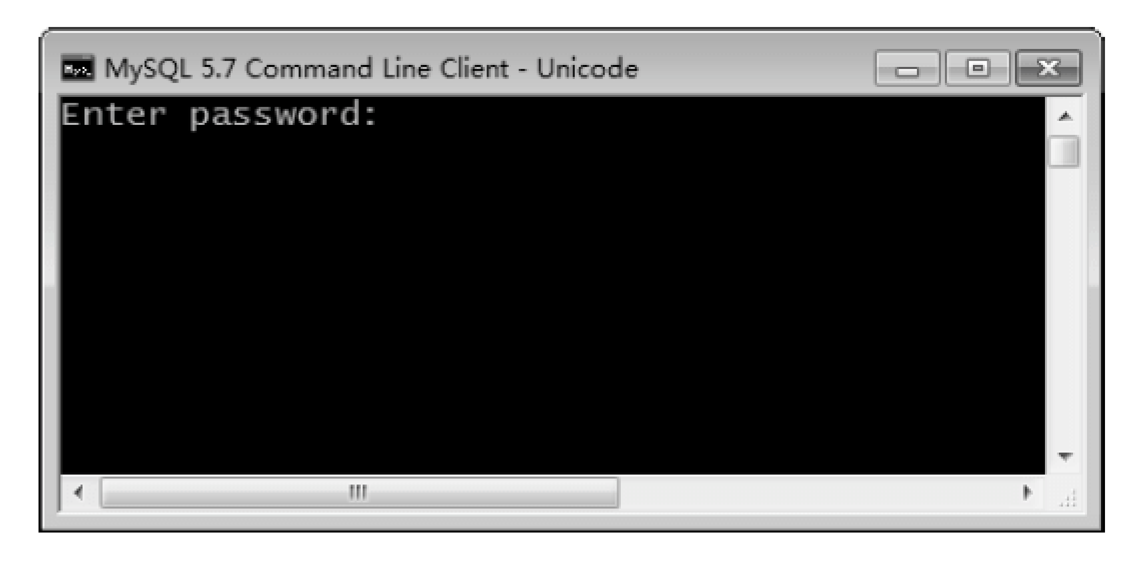


图 14-8　密码输入窗口

输入正确的密码之后，就可以登录到 MySQL 数据库了。

14.2.3 配置 Path 变量

在前面登录 MySQL 服务器的时候，不能直接输入 MySQL 登录命令，因为没有把 MySQL 的 bin 目录添加到系统的环境变量里面，所以不能直接使用 MySQL 命令。如果每次登录都输入“cd C:\Program Files\MySQL\MySQL Server 5.7\bin”，才能使用 MySQL 等其他命令工具，这样比较麻烦。

下面介绍如何手动配置 PATH 变量。具体的操作步骤如下。

01 在桌面上右击【我的电脑】图标，在弹出的快捷菜单中选择【属性】菜单命令，如图 14-9 所示。

02 在打开的【系统】窗口中，单击【高级系统设置】选项，如图 14-10 所示。



图 14-9 【我的电脑】属性菜单



图 14-10 【系统】窗口

03 打开【系统属性】对话框，并选择【高级】选项卡，如图 14-11 所示。

04 单击【环境变量】按钮，打开【环境变量】对话框，在【系统变量】列表中选择 Path 变量，如图 14-12 所示。



图 14-11 【系统属性】对话框



图 14-12 【环境变量】对话框

05 单击【编辑】按钮，在【编辑系统变量】对话框中，将MySQL应用程序的bin目录（C:\Program Files\MySQL\MySQL Server 5.7\bin）添加到变量值中，用分号将其与其他路径分隔开，如图 14-13 所示。



图 14-13 【编辑系统变量】对话框

06 添加完成之后，单击【确定】按钮，这样就完成了配置 PATH 变量的操作，然后就可以直接输入 MySQL 命令来登录数据库了。

14.3 MySQL 常用图形管理工具

MySQL 图形化管理工具极大地方便了数据库的操作与管理，常用的图形化管理工具有：MySQL Workbench、phpMyAdmin、Navicat、MySQLDumper、SQLyog、MySQL ODBC Connector。其中 phpMyAdmin 和 Navicat 提供中文操作界面；MySQL Workbench、MySQL ODBC Connector、MySQLDumper 为英文界面。下面介绍几个常用的图形管理工具。

1. MySQL Workbench

MySQL 官方提供的图形化管理工具 MySQl Workbench 完全支持 MySQL 5.0 以上的版本，在 5.0 版本中，有些功能将不能使用；而在 4.X 以下的版本中，MySQL Workbench 分为社区版和商业版，社区版完全免费，而商业版则是按年收费。

下载地址：http://dev.MySQL.com/downloads/workbench/。

2. phpMyAdmin

phpMyAdmin 使用 PHP 编写，必须安装在 Web 服务器中，通过 Web 方式控制和操作 MySQL 数据库。通过 phpMyAdmin 可以完全对数据库进行操作，例如建立、复制、删除数据等等。管理数据库非常方便，并支持中文，不足之处在于对大数据库的备份和恢复不方便。

下载地址：http://www.phpmyadmin.net/。

3. MySQLDumper

MySQLDumper 使用基于 PHP 开发的 MySQL 数据库备份恢复程序，解决了使用 PHP 进行大数据库备份和恢复的问题。数百兆的数据库都可以方便地备份恢复，不用担心网速太慢而导致中断的问题，非常方便易用。

下载地址：http://www.MySQLdumper.de/en/。

14.4 高手私房菜

计算机技术具有很强的操作性，MySQL 的安装和配置是一件非常简单的事，但是在操作过程中也可能出现问题，读者需要多实践、多总结。

疑问 1：无法打开 MySQL 5.7 软件安装包，提示对话框如图 14-14 所示，如何解决？

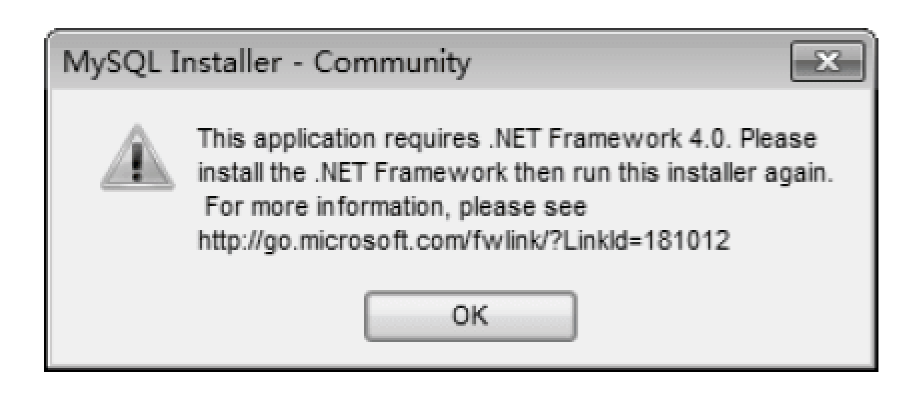


图 14-14 无法安装提示对话框

在安装MySQL 5.7软件安装包之前，用户需要确保系统中已经安装了.Net Framework 3.5和.Net Framework 4.0，如果缺少这两个软件，将不能正常地安装 MySQL 5.7 软件。另外还要确保的WindowsInstaller 正常安装。

疑问 2：MySQL 安装失败？

安装过程失败，多数是由于重新安装 MySQL 的缘故，因为 MySQL 在删除的时候，不能自动删除相关的信息。解决方法是，把以前安装的目录删除掉。删除在 C 盘的 program file 文件夹里面MySQL 的安装目录文件夹；同时删除 MySQL 的 DATA 目录，该目录一般为隐藏文件，其位置一般在“C:\Documents and Settings\All Users\Application Data\ MySQL”目录下，删除掉后重新安装即可。

14.5 经典习题

（1）下载并安装 MySQL。

（2）使用配置向导配置 MySQL 为系统服务，在系统服务对话框中，手动启动或者关闭MySQL 服务。

（3）使用 net 命令启动或者关闭 MySQL 服务。

（4）使用免安装的软件包安装 MySQL。

第 15 章　数据库的基本操作

MySQL 安装好以后，首先需要创建数据库，这是使用 MySQL 各种功能的前提。本章将详细介绍数据的基本操作，主要内容包括：创建数据库、删除数据库等。

本章学习目标

- 掌握如何创建数据库
- 熟悉数据库的删除操作
- 掌握综合案例中数据库的创建和删除方法

15.1　创建数据库

MySQL 安装完成之后，将会在其 data 目录下自动创建几个必需的数据库，可以使用 SHOW DATABASES;语句来查看当前所有存在的数据库，输入语句如下。

```
mysql> SHOW DATABASES;
+-----------------------+
| Database              |
+-----------------------+
| information_schema  |
| mysql                |
| performance_schema  |
| sakila               |
| test                  |
| world                     |
+--------------------------+
6 rows in set (0.04 sec)
```

可以看到，数据库列表中包含了 6 个数据库，其中 MySQL 是必需的，它描述用户访问权限，用户经常利用 test 数据库做测试的工作，其他数据库在这里不做介绍。

创建数据库是在系统磁盘上划分一块区域用于数据的存储和管理，如果管理员在设置权限的时候为用户创建了数据库，则可以直接使用，否则，需要自己创建数据库。MySQL 中创建数据库的基本 SQL 语法格式为：

```
CREATE DATABASE database_name;
```

“database_name”为要创建的数据库的名称，该名称不能与已经存在的数据库重名。

【例 15.1】创建测试数据库 test_db，输入语句如下。

```
CREATE DATABASE test_db;
```

数据库创建好之后，可以使用 SHOW CREATE DATABASE 声明查看数据库的定义。

【例 15.2】查看创建好的数据库 test_db 的定义，输入语句如下。

```
mysql> SHOW CREATE DATABASE test_db\G
*** 1. row ***
       Database: test_db
Create Database: CREATE DATABASE `test_db` /*!40100 DEFAULT CHARACTER SET utf8 */
1 row in set (0.00 sec)
```

可以看到，如果数据库创建成功，将显示数据库的创建信息。

再次使用 SHOW DATABASES;语句来查看当前所有存在的数据库，输入语句如下。

```
mysql> SHOW databases;
+-------------------------+
| Database          |
+-------------------------+
| information_schema |
| mysql            |
| performance_schema |
| sakila              |
| test              |
| test_db           |
| world               |
+------------------------+
7 rows in set (0.05 sec)
```

可以看到，数据库列表中包含了刚刚创建的数据库 test_db 和其他已经存在的数据库的名称。

15.2 删除数据库

删除数据库是将已经存在的数据库从磁盘空间上清除，清除之后，数据库中的所有数据也将一同被删除。删除数据库语句和创建数据库的命令相似，MySQL 中删除数据库的基本语法格式为：

```
DROP DATABASE database_name;
```

“database_name”为要删除的数据库的名称，如果指定的数据库不存在，则删除出错。

【例 15.3】删除测试数据库 test_db，输入语句如下。

```
DROP DATABASE test_db;
```

语句执行完毕之后，数据库 test_db 将被删除，再次使用 SHOW CREATE DATABASE 声明查看数据库的定义，结果如下。

```
mysql> SHOW CREATE DATABASE test_db\G
ERROR 1049 (42000): Unknown database 'test_db'
ERROR:
No query specified
```

执行结果给出一条错误信息："ERROR 1049 (42000): Unknown database 'test_db'"，即数据库 test_db 已不存在，删除成功。


提 示

使用 DROP DATABASE 命令时要非常谨慎，在执行该命令时，MySQL 不会给出任何提醒确认信息，DROP DATABASE 声明删除数据库后，数据库中存储的所有数据表和数据也将一同被删除，而且不能恢复。

15.3 实战演练——数据库的创建和删除

本章分别介绍了数据库的基本操作，包括数据库的创建、查看当前数据库和删除数据库。最后介绍了 MySQL 中各种存储引擎。在这里，通过一个案例，让读者全面回顾数据库的基本操作。

1. 案例目的

登录 MySQL，使用数据库操作语句创建、查看和删除数据库，步骤如下：

（1）登录数据库。
（2）创建数据库 zoo。
（3）选择当前数据库为 zoo，并查看 zoo 数据库的信息。
（4）删除数据库 zoo。

2. 案例操作过程

01 登录数据库。

打开 Windows 命令行，输入登录用户名和密码。

```
C:\>mysql -h localhost -u root -p
Enter password: **
```

或者打开 MySQL 5.7 Command Line Client，只用输入用户密码也可以登录。登录成功后显示如下信息：

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.10 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

出现 MySQL 命令输入提示符时表示登录成功，可以输入 SQL 语句进行操作。

02 创建数据库 zoo，执行过程如下。

```
mysql> CREATE DATABASE zoo;
Query OK, 1 row affected (0.01 sec)
```

提示信息表明语句成功执行。

查看当前系统中所有的数据库，执行过程如下。

```
mysql> SHOW DATABASES;
+-------------------------+
| Database          |
+------------------------+
| information_schema |
| mysql            |
| performance_schema |
| test              |
| sakila             |
| test               |
| world             |
| zoo               |
+-------------------------+
```

可以看到，数据库列表中已经有了名称为 zoo 数据库，数据库创建成功。

03 选择当前数据库为 zoo，查看数据库 zoo 的信息，执行过程如下。

```
mysql> USE zoo;
Database changed
```

提示信息 Database changed 表明选择成功。

查看数据库信息，如下所示：

```
mysql> SHOW CREATE DATABASE zoo \G
*** 1. row ***
      Database: zoo
Create Database: CREATE DATABASE `zoo` /*!40100 DEFAULT CHARACTER SET utf8 */
```

Database 值表明当前数据库名称；Create Database 值表示创建数据库 zoo 的语句，后面为注释信息。

04 删除数据库 zoo，执行过程如下。

```
mysql> DROP DATABASE zoo;
Query OK, 0 rows affected (0.00 sec)
```

语句执行完毕，将数据库 zoo 从系统中删除。

```
mysql> SHOW DATABASES;
+------------------------+
| Database          |
+------------------------+
| information_schema |
| mysql            |
```

```
| performance_schema |
| test               |
| sakila             |
| test               |
| world              |
+--------------------------+
```

可以看到，数据库列表中已经没有名称为 zoo 的数据库了。

15.4 高手私房菜

技巧 1：什么是 MySQL 存储引擎？

MySQL 提供了多个不同的存储引擎，包括处理事务安全表的引擎和处理非事务安全表的引擎。在 MySQL 中，不需要在整个服务器中使用同一种存储引擎，针对具体的要求，可以对每一个表使用不同的存储引擎。MySQL 5.7 支持的存储引擎有：InnoDB，MyISAM，Memory，Merge，Archive，Federated，CSV，BLACKHOLE 等。可以使用 SHOW ENGINES 语句查看系统所支持的引擎类型，结果如下。

```
mysql> SHOW ENGINES \G
*** 1. row ***
      Engine: FEDERATED
     Support: NO
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
*** 2. row ***
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*** 3. row ***
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*** 4. row ***
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
          XA: NO
  Savepoints: NO
```

```
*** 5. row ***
      Engine: CSV
     Support: YES
     Comment: CSV storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*** 6. row ***
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
  Savepoints: NO
*** 7. row ***
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*** 8. row ***
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*** 9. row ***
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
9 rows in set (0.00 sec)
```

Support 列的值表示某种引擎是否能使用：YES 表示可以使用，NO 表示不能使用，DEFAULT 表示该引擎为当前默认存储引擎。

15.5 经典习题

（1）查看当前系统中的数据库。

（2）创建数据库 Book，使用 SHOW CREATE DATABASE 语句查看数据库定义信息。

（3）删除数据库 Book。

第 16 章　数据表的基本操作

在数据库中，数据表是数据库中最重要、最基本的操作对象，是数据存储的基本单位。数据表被定义为列的集合，数据在表中是按照行和列的格式来存储的。每一行代表一条唯一的记录，每一列代表记录中的一个域。

本章将详细介绍数据表的基本操作，主要内容包括：创建数据表、查看数据表结构、修改数据表、删除数据表。通过本章的学习，读者能够熟练掌握数据表的基本概念，理解约束、默认和规则的含义并且学会运用；能够在图形界面模式和命令行模式下熟练地完成有关数据表的常用操作。

本章学习目标

- 掌握如何创建数据表
- 掌握查看数据表结构的方法
- 掌握如何修改数据表
- 熟悉删除数据表的方法
- 熟练操作综合案例数据表的基本操作

16.1　创建数据表

在创建完数据库之后，接下来的工作就是创建数据表。所谓创建数据表，指的是在已经创建好的数据库中建立新表。创建数据表的过程是规定数据列的属性的过程，同时也是实施数据完整性（包括实体完整性、引用完整性和域完整性等）约束的过程。本节将介绍创建数据表的语法形式、如何添加主键约束、外键约束、非空约束等。

16.1.1　创建表的语法形式

数据表属于数据库，在创建数据表之前，应该使用语句“USE <数据库名>”指定操作是在哪个数据库中进行，如果没有选择数据库，会抛出“No database selected”的错误。

创建数据表的语句为 CREATE TABLE，语法规则如下：

```
CREATE  TABLE <表名>
(
字段名1，数据类型 [列级别约束条件] [默认值]，
字段名2，数据类型 [列级别约束条件] [默认值]，
……
[表级别约束条件]
);
```

使用 CREATE TABLE 创建表时，必须指定以下信息：

（1）要创建的表的名称，不区分大小写，不能使用 SQL 语言中的关键字，如 DROP、ALTER、INSERT 等。

（2）数据表中每一个列（字段）的名称和数据类型，如果创建多个列，要用逗号隔开。

【例 16.1】创建员工表 tb_emp1，结构如表 16-1 所示。

表 16-1　tb_emp1 表结构

字段名称	数据类型	备注
id	INT(11)	员工编号
name	VARCHAR(25)	员工名称
deptId	INT(11)	所在部门编号
salary	FLOAT	工资

首先创建数据库，SQL 语句如下：

```
CREATE  DATABASE test_db;
```

选择创建表的数据库，SQL 语句如下：

```
USE test_db;
```

创建 tb_emp1 表，SQL 语句为：

```
CREATE TABLE tb_emp1
(
id      INT(11),
name    VARCHAR(25),
deptId  INT(11),
salary  FLOAT
);
```

语句执行后，便创建了一个名称为 tb_emp1 的数据表，使用 SHOW TABLES;语句查看数据表是否创建成功，SQL 语句如下：

```
mysql> SHOW TABLES;
+-----------------------+
| Tables_in_ test_db |
+----------------------+
| tb_emp1          |
+----------------------+
1 row in set (0.00 sec)
```

可以看到，test_db 数据库中已经有了数据表 tb_tmp1，数据表创建成功。

16.1.2　使用主键约束

主键，又称主码，是表中一列或多列的组合。主键约束（Primary Key Constraint）要求主键列的数据唯一，并且不允许为空。主键能够惟一地标识表中的一条记录，可以结合外键来定义不同数

据表之间的关系，并且可以加快数据库查询的速度。主键和记录之间的关系如同身份证和人之间的关系，它们之间是一一对应的。主键分为两种类型：单字段主键和多字段联合主键。

1. 单字段主键

主键由一个字段组成，SQL 语句格式分为以下两种情况。

（1）在定义列的同时指定主键，语法规则如下：

```
字段名 数据类型 PRIMARY KEY [默认值]
```

【例 16.2】定义数据表 tb_emp 2，其主键为 id，SQL 语句如下：

```
CREATE TABLE tb_emp2
(
id        INT(11) PRIMARY KEY,
name      VARCHAR(25),
deptId    INT(11),
salary    FLOAT
);
```

（2）在定义完所有列之后指定主键。

```
[CONSTRAINT <约束名>] PRIMARY KEY [字段名]
```

【例 16.3】定义数据表 tb_emp 3，其主键为 id，SQL 语句如下：

```
CREATE TABLE tb_emp3
(
id INT(11),
name VARCHAR(25),
deptId INT(11),
salary FLOAT,
PRIMARY KEY(id)
);
```

上述两个例子执行后的结果是一样的，都会在 id 字段上设置主键约束。

2. 多字段联合主键

主键由多个字段联合组成，语法规则如下：

```
PRIMARY KEY [字段1, 字段2, . . ., 字段 n]
```

【例 16.4】定义数据表 tb_emp4，假设表中间没有主键 id，为了唯一确定一个员工，可以把 name、deptId 联合起来做为主键，SQL 语句如下：

```
CREATE TABLE tb_emp4
 (
name VARCHAR(25),
deptId INT(11),
salary FLOAT,
```

```
PRIMARY KEY(name,deptId)
);
```

语句执行后，便创建了一个名称为 tb_emp4 的数据表，name 字段和 deptId 字段组合在一起成为 tb_emp4 的多字段联合主键。

16.1.3 使用外键约束

外键用来在两个表的数据之间建立链接，它可以是一列或者多列。一个表可以有一个或多个外键。外键对应的是参照完整性，一个表的外键可以为空值，若不为空值，则每一个外键值必须等于另一个表中主键的某个值。

外键：首先它是表中的一个字段，它可以不是本表的主键，但对应另外一个表的主键。外键主要作用是保证数据引用的完整性，定义外键后，不允许删除在另一个表中具有关联关系的行。外键的作用是保持数据的一致性、完整性。例如，部门表 tb_dept 的主键是 id，在员工表 tb_emp5 中有一个键 deptId 与这个 id 关联。

主表（父表）：对于两个具有关联关系的表而言，相关联字段中主键所在的那个表即是主表。

从表（子表）：对于两个具有关联关系的表而言，相关联字段中外键所在的那个表即是从表。

创建外键的语法规则如下：

```
[CONSTRAINT <外键名>] FOREIGN KEY 字段名1 [ ,字段名2,…]
REFERENCES <主表名> 主键列1 [ ,主键列2,…]
```

“外键名”为定义的外键约束的名称，一个表中不能有相同名称的外键；“字段名”表示子表需要添加外键约束的字段列；“主表名”即被子表外键所依赖的表的名称；“主键列”表示主表中定义的主键列，或者列组合。

【例 16.5】定义数据表 tb_emp5，并在 tb_emp5 表上创建外键约束。

创建一个部门表 tb_dept1，表结构如表 16-2 所示，SQL 语句如下：

表 16-2 tb_dept1 表结构

字段名称	数据类型	备注
id	INT(11)	部门编号
name	VARCHAR(22)	部门名称
location	VARCHAR(50)	部门位置

```
CREATE TABLE tb_dept1
(
id       INT(11) PRIMARY KEY,
name     VARCHAR(22) NOT NULL,
location  VARCHAR(50)
);
```

定义数据表 tb_emp5，让它的键 deptId 作为外键关联到 tb_dept1 的主键 id，SQL 语句为：

```
CREATE TABLE tb_emp5
```

```
(
id      INT(11) PRIMARY KEY,
name    VARCHAR(25),
deptId  INT(11),
salary   FLOAT,
CONSTRAINT fk_emp_dept1 FOREIGN KEY(deptId) REFERENCES tb_dept1(id)
);
```

以上语句执行成功之后，在表 tb_emp5 上添加了名称为 fk_emp_dept1 的外键约束，外键名称为 deptId，其依赖于表 tb_dept1 的主键 id。


提 示

关联指的是在关系型数据库中，相关表之间的联系。它是通过相容或相同的属性或属性组来表示的。子表的外键必须关联父表的主键，且关联字段的数据类型必须匹配，如果类型不一样，则创建子表时，就会出现错误“ERROR 1005 (HY000): Can't create table 'database.tablename'(errno: 150)”。

16.1.4 使用非空约束

非空约束（Not Null Constraint）指字段的值不能为空。对于使用了非空约束的字段，如果用户在添加数据时没有指定值，数据库系统会报错。

非空约束的语法规则如下：

```
字段名 数据类型 not null
```

【例 16.6】定义数据表 tb_emp6，指定员工的名称不能为空，SQL 语句如下：

```
CREATE TABLE tb_emp6
(
id     INT(11) PRIMARY KEY,
name    VARCHAR(25) NOT NULL,
deptId  INT(11),
salary  FLOAT
);
```

执行后，在 tb_emp6 中创建了一个 Name 字段，其插入值不能为空（NOT NULL）。

16.1.5 使用唯一性约束

唯一性约束（Unique Constraint）要求该列唯一，允许为空，但只能出现一个空值。唯一性约束可以确保一列或者几列不出现重复值。

唯一性约束的语法规则如下：

（1）在定义完列之后直接指定唯一约束，语法规则如下：

```
字段名 数据类型 UNIQUE
```

【例 16.7】定义数据表 tb_dept2，指定部门的名称唯一，SQL 语句如下：

```
CREATE TABLE tb_dept2
(
id      INT(11) PRIMARY KEY,
name    VARCHAR(22) UNIQUE,
location  VARCHAR(50)
);
```

（2）在定义完所有列之后指定唯一约束，语法规则如下：

```
[CONSTRAINT <约束名>] UNIQUE(<字段名>)
```

【例 16.8】定义数据表 tb_dept3，指定部门的名称唯一，SQL 语句如下：

```
CREATE TABLE tb_dept3
(
id      INT(11) PRIMARY KEY,
name    VARCHAR(22),
location  VARCHAR(50),
CONSTRAINT STH UNIQUE(name)
);
```

UNIQUE 和 PRIMARY KEY 的区别：一个表中可以有多个字段声明为 UNIQUE，但只能有一个 PRIMARY KEY 声明；声明为 PRIMAY KEY 的列不允许有空值，但是声明为 UNIQUE 的字段允许空值（NULL）的存在。

16.1.6 使用默认约束

默认约束（Default Constraint）指定某列的默认值。如男性同学较多，性别就可以默认为‘男’。如果插入一条新的记录时没有为这个字段赋值，那么系统会自动为这个字段赋值为‘男’。

默认约束的语法规则如下：

```
字段名 数据类型 DEFAULT 默认值
```

【例 16.9】定义数据表 tb_emp7，指定员工的部门编号默认为 1111，SQL 语句如下：

```
CREATE TABLE tb_emp7
(
id      INT(11) PRIMARY KEY,
name   VARCHAR(25) NOT NULL,
deptId  INT(11) DEFAULT 1111,
salary  FLOAT
);
```

以上语句执行成功之后，表 tb_emp7 上的字段 deptId 拥有了一个默认的值 1111，新插入的记录如果没有指定部门编号，则默认都为 1111。

16.1.7 设置表的属性值自动增加

在数据库应用中，经常希望在每次插入新记录时，系统自动生成字段的主键值。可以通过为表主键添加 AUTO_INCREMENT 关键字来实现。默认的，在 MySQL 中 AUTO_INCREMENT 的初始值值是 1，每新增一条记录，字段值自动加 1。一个表只能有一个字段使用 AUTO_INCREMENT 约束，且该字段必须为主键的一部分。AUTO_INCREMENT 约束的字段可以是任何整数类型（TINYINT、SMALLIN、INT、BIGINT 等）。

设置表的属性值自动增加的语法规则如下：

```
字段名 数据类型 AUTO_INCREMENT
```

【例 16.10】定义数据表 tb_emp8，指定员工的编号自动递增，SQL 语句如下：

```
CREATE TABLE tb_emp8
(
id      INT(11) PRIMARY KEY AUTO_INCREMENT,
name    VARCHAR(25) NOT NULL,
deptId  INT(11),
salary  FLOAT
);
```

上述例子执行后，会创建名称为 tb_emp8 的数据表。表 tb_emp8 中的 id 字段的值在添加记录的时候会自动增加，在插入记录的时候，默认的自增字段 id 的值从 1 开始，每次添加一条新记录，该值自动加 1。

例如，执行如下插入语句：

```
mysql> INSERT INTO tb_emp8 (name,salary)
-> VALUES('Lucy',1000), ('Lura',1200),('Kevin',1500);
```

语句执行完后，tb_emp8 表中增加 3 条记录，在这里并没有输入 id 的值，但系统已经自动添加该值，使用 SELECT 命令查看记录，如下所示。

```
mysql> SELECT * FROM tb_emp8;
+----+--------+----------+------------+
| id | name   | deptId| salary      |
+----+--------+-----------+------------+
|  1 | Lucy   | NULL |   1000   |
|  2 | Lura   | NULL |   1200   |
|  3 | Kevin  | NULL |   1500   |
+----+--------+-----------+------------+
3 rows in set (0.00 sec)
```


提 示

这里使用 INSERT 声明向表中插入记录的方法，并不是 SQL 的标准语法，这种语法不一定被其他的数据库支持，只能在 MySQL 中使用。

16.2 查看数据表结构

使用 SQL 语句创建好数据表之后，可以查看表结构的定义，以确认表的定义是否正确。在 MySQL 中，查看表结构可以使用 DESCRIBE 和 SHOW CREATE TABLE 语句。本节将针对这两个语句分别进行详细的讲解。

16.2.1 查看表基本结构语句 DESCRIBE

DESCRIBE/DESC 语句可以查看表的字段信息，其中包括：字段名、字段数据类型、是否为主键、是否有默认值等。语法规则如下：

```
DESCRIBE 表名;
```

或者简写为：

```
DESC 表名;
```

【例 16.11】分别使用 DESCRIBE 和 DESC 查看表 tb_dept1 和表 tb_emp1 的表结构。

查看 tb_dept1 表结构，SQL 语句如下：

```
mysql> DESCRIBE tb_dept1;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| id         | int(11)       | NO   | PRI | NULL    |       |
| name       | varchar(22)   | NO   |     | NULL    |       |
| location   | varchar(50)   | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
```

查看 tb_emp1 表结构，SQL 语句如下：

```
mysql> DESC tb_emp1;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int (11)    | YES  |     | NULL    |       |
| name   | varchar(25) | YES  |     | NULL    |       |
| deptId | int (11)    | YES  |     | NULL    |       |
| salary | float       | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

其中，各个字段的含义分别解释如下：

- Null：表示该列是否可以存储 Null 值。
- Key：表示该列是否已编制索引。PRI 表示该列是表主键的一部分；UNI 表示该列是 UNIQUE 索引的一部分；MUL 表示在列中某个给定值允许出现多次。
- Default：表示该列是否有默认值，如果有的话值是多少。
- Extra：表示可以获取的与给定列有关的附加信息，例如 AUTO_INCREMENT 等。

16.2.2　查看表详细结构语句 SHOW CREATE TABLE

SHOW CREATE TABLE 语句可以用来显示创建表时的 CREATE TABLE 语句，语法格式如下：

```
SHOW CREATE TABLE <表名\G>;
```



提　示

使用 SHOW CREATE TABLE 语句，不仅可以查看表创建时候的详细语句，而且还可以查看存储引擎和字符编码。

如果不加‘\G’参数，显示的结果可能非常混乱，加上参数‘\G’之后，可使显示结果更加直观，易于查看。

【例 16.12】使用 SHOW CREATE TABLE 查看表 tb_emp1 的详细信息，SQL 语句如下：

```
mysql> SHOW CREATE TABLE tb_emp1;
+--------+-----------------------------------------------------------------
-----------------------------------
 -----------------------------------------------------------------
-----------------------------------
 -------------------------------------------------------------+
| Table  | Create Table

                                                    |
+--------+-----------------------------------------------------------------
-----------------------------------
 -----------------------------------------------------------------
-----------------------------------
 -------------------------------------------------------------+
| fruits | CREATE TABLE `fruits` (
  `f_id` char(10) NOT NULL,
  `s_id` int(11) NOT NULL,
  `f_name` char(255) NOT NULL,
  `f_price` decimal(8,2) NOT NULL,
  PRIMARY KEY (`f_id`),
  KEY `index_name` (`f_name`),
  KEY `index_id_price` (`f_id`,`f_price`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312 |
+--------+-----------------------------------------------------------------
-----------------------------------
 -----------------------------------------------------------------
-----------------------------------
 -------------------------------------------------------------+
```

使用参数‘\G’之后的结果如下：

```
mysql> SHOW CREATE TABLE tb_emp1\G
```

```
*** 1. row ***
      Table: tb_emp1
Create Table: CREATE TABLE `tb_emp1` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

16.3　修改数据表

修改表指的是修改数据库中已经存在的数据表的结构。MySQL 使用 ALTER TABLE 语句修改表。常用的修改表的操作有：修改表名、修改字段数据类型或字段名、增加和删除字段、修改字段的排列位置，更改表的存储引擎，删除表的外键约束等。本节将对和修改表有关的操作进行讲解。

16.3.1　修改表名

MySQL 是通过 ALTER TABLE 语句来实现表名的修改的，具体的语法规则如下：

```
ALTER TABLE <旧表名> RENAME [TO] <新表名>;
```

其中 TO 为可选参数，使用与否均不影响结果。

【例 16.13】将数据表 tb_dept3 改名为 tb_deptment3。

执行修改表名操作之前，使用 SHOW TABLES 查看数据库中所有的表。

```
mysql> SHOW TABLES;
+---------------------+
| Tables_in_test_db   |
+--------------------+
| tb_dept            |
| tb_dept2           |
| tb_dept3           |
```

使用 ALTER TABLE 将表 tb_dept3 改名为 tb_deptment3，SQL 语句如下：

```
ALTER TABLE tb_dept3 RENAME tb_deptment3;
```

语句执行之后，检验表 tb_dept3 是否改名成功。使用 SHOW TABLES 查看数据库中的表，结果如下：

```
mysql> SHOW TABLES;
+---------------------+
| Tables_in_test_db   |
+---------------------+
| tb_dept             |
| tb_dept2            |
```

```
| tb_deptment3          |
```

经过比较可以看到，数据表列表中已经有了名称为 tb_deptment3 的表。


提 示

读者可以在修改表名称时使用 DESC 命令查看修改前后两个表的结构，修改表名并不修改表的结构，因此修改名称后的表和修改名称前的表的结构必然是相同的。

16.3.2 修改字段的数据类型

修改字段的数据类型，就是把字段的数据类型转换成另一种数据类型。在 MySQL 中修改字段数据类型的语法规则如下：

```
ALTER TABLE <表名> MODIFY <字段名>  <数据类型>
```

其中“表名”指要修改数据类型的字段所在表的名称，“字段名”指需要修改的字段，“数据类型”指修改后字段的新数据类型。

【例 16.14】将数据表 tb_dept1 中 name 字段的数据类型由 VARCHAR(22)修改成 VARCHAR(30)。

执行修改表名操作之前，使用 DESC 查看 tb_dept 表结构，结果如下：

```
mysql> DESC tb_dept1;
+----------+---------------+---------+--------+-------------+-------+
| Field    | Type          | Null    | Key    |Default      | Extra |
+----------+---------------+---------+--------+-------------+-------+
| id       | int(11)       | NO      | PRI    | NULL        |       |
| name     | varchar(22)   | YES     |        | NULL        |       |
| location | varchar(50)   | YES     |        | NULL        |       |
+----------+---------------+--------+---------+-------------+-------+
3 rows in set (0.00 sec)
```

可以看到现在 name 字段的数据类型为 VARCHAR(22)，下面修改其类型。输入如下 SQL 语句并执行：

```
ALTER TABLE tb_dept1 MODIFY name VARCHAR(30);
```

再次使用 DESC 查看表，结果如下：

```
mysql> DESC tb_dept1;
+----------+---------------+---------+--------+-------------+-------+
| Field    | Type          | Null    | Key    |Default      | Extra |
+----------+---------------+---------+--------+-------------+-------+
| id       | int(11)       | NO      | PRI    | NULL        |       |
| name     | varchar(30)   | YES     |        | NULL        |       |
| location | varchar(50)   | YES     |        | NULL        |       |
+----------+---------------+--------+---------+-------------+-------+
3 rows in set (0.00 sec)
```

语句执行之后，检验会发现表tb_dept表中name字段的数据类型已经修改成了VARCHAR(30)，修改成功。

16.3.3　修改字段名

MySQL 中修改表字段名的语法规则如下：

```
ALTER TABLE <表名> CHANGE <旧字段名> <新字段名> <新数据类型>;
```

其中，“旧字段名”指修改前的字段名；“新字段名”指修改后的字段名；“新数据类型”指修改后的数据类型，如果不需要修改字段的数据类型，可以将新数据类型设置成与原来一样即可，但数据类型不能为空。

【例 16.15】将数据表 tb_dept1 中的 location 字段名称改为 loc，数据类型保持不变，SQL 语句如下：

```
ALTER TABLE tb_dept1 CHANGE location loc VARCHAR(50);
```

使用 DESC 查看表 tb_dept1，会发现字段的名称已经修改成功，结果如下：

```
mysql> DESC tb_dept1;
+----------+--------------+---------+--------+-------------+-------+
| Field    | Type         | Null    | Key    |Default      | Extra |
+----------+--------------+---------+--------+-------------+-------+
| id       | int(11)      | NO      | PRI    | NULL        |       |
| name     | varchar(30)  | YES     |        | NULL        |       |
| loc      | varchar(50)  | YES     |        | NULL        |       |
+----------+--------------+--------+---------+-------------+-------+
3 rows in set (0.00 sec)
```

【例 16.16】将数据表 tb_dept1 中的 loc 字段名称改为 location，同时将数据类型变为 VARCHAR(60)，SQL 语句如下：

```
ALTER TABLE tb_dept1 CHANGE loc location VARCHAR(60);
```

使用 DESC 查看表 tb_dept1，会发现字段的名称和数据类型均已经修改成功，结果如下：

```
mysql> DESC tb_dept1;
+----------+--------------+---------+--------+-------------+-------+
| Field    | Type         | Null    | Key    |Default      | Extra |
+----------+--------------+---------+--------+-------------+-------+
| id       | int(11)      | NO      | PRI    | NULL        |       |
| name     | varchar(30)  | YES     |        | NULL        |       |
| location | varchar(60)  | YES     |        | NULL        |       |
+----------+--------------+--------+---------+-------------+-------+
3 rows in set (0.00 sec)
```



提 示

CHANGE 也可以只修改数据类型，实现和 MODIFY 同样的效果，方法是将 SQL 语句中的“新字段名”和“旧字段名”设置为相同的名称，只改变“数据类型”。

由于不同类型的数据在机器中存储的方式及长度并不相同，修改数据类型可能会影响到数据表中已有的数据记录。因此，当数据库表中已经有数据时，不要轻易修改数据类型。

16.3.4 添加字段

随着业务需求的变化，可能需要在已经存在的表中添加新的字段。一个完整字段包括字段名、数据类型、完整性约束。添加字段的语法格式如下：

```
ALTER TABLE <表名> ADD <新字段名> <数据类型>
[约束条件] [FIRST | AFTER 已存在字段名];
```

新字段名为需要添加的字段的名称；“FIRST”为可选参数，其作用是将新添加的字段设置为表的第一个字段；“AFTER”为可选参数，其作用是将新添加的字段添加到指定的“已存在字段名”的后面。


提 示

“FIRST”或“AFTER 已存在字段名”用于指定新增字段在表中的位置，如果 SQL 语句中没有这两个参数，则默认将新添加的字段设置为数据表的最后列。

1. 添加无完整性约束条件的字段

【例 16.17】在数据表 tb_dept1 中添加一个没有完整性约束的 INT 类型的字段 managerId（部门经理编号），SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD managerId INT(10);
```

使用 DESC 查看表 tb_dept1，会发现在表的最后添加了一个名为 managerId 的 INT 类型的字段，结果如下：

```
mysql> DESC tb_dept1;
+-------------+----------------+----------+--------+-------------+--------
+
| Field     | Type        | Null    | Key    | Default   | Extra  |
+-------------+----------------+----------+--------+-------------+--------
+
| id         | int(11)       | NO     | PRI    | NULL    |        |
| name      | varchar(30)  | YES     |        | NULL    |        |
| location    | varchar(60)  | YES     |        | NULL    |        |
| managerId | int(10)       | YES     |        | NULL    |        |
+-------------+----------------+----------+--------+-------------+--------
+
4 rows in set (0.03 sec)
```

2. 添加有完整性约束条件的字段

【例 16.18】在数据表 tb_dept1 中添加一个不能为空的 VARCHAR(12)类型的字段 column1，SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD column1 VARCHAR(12) not null;
```

使用 DESC 查看表 tb_dept1，会发现在表的最后添加了一个名为 column1 的 VARCHAR(12)类型且不为空的字段，结果如下：

```
mysql> DESC tb_dept1;
+-----------+----------------+----------+---------+------------+--------+
| Field     | Type           | Null     | Key     | Default    | Extra |
+-----------+----------------+----------+---------+------------+--------+
| id        | int(11)        | NO       | PRI     | NULL       |       |
| name      | varchar(30)    | YES      |         | NULL       |       |
| location  | varchar(60)    | YES      |         | NULL       |       |
| managerId | int(10)        | YES      |         | NULL       |       |
| column1   | varchar(12)    | NO       |         | NULL       |       |
+-----------+----------------+----------+---------+------------+--------+
5 rows in set (0.00 sec)
```

3. 在表的第一列添加一个字段

【例 16.19】在数据表 tb_dept1 中添加一个 INT 类型的字段 column2，SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD column2 INT(11) FIRST;
```

使用 DESC 查看表 tb_dept1，会发现在表第一列添加了一个名为 column2 的 INT(11)类型字段，结果如下：

```
mysql> DESC tb_dept1;
+------------+---------------+----------+---------+------------+--------+
| Field      | Type          | Null     | Key     | Default    | Extra  |
+------------+---------------+----------+---------+------------+--------+
| column2    | int(11)       | YES      |         | NULL       |        |
| id         | int(11)       | NO       | PRI     | NULL       |        |
| name       | varchar(30)   | YES      |         | NULL       |        |
| location   | varchar(60)   | YES      |         | NULL       |        |
| managerId  | int(10)       | YES      |         | NULL       |        |
| column1    | varchar(12)   | NO       |         | NULL       |        |
+------------+---------------+----------+---------+------------+--------+
6 rows in set (0.00 sec)
```

4. 在表的指定列之后添加一个字段

【例 16.20】在数据表 tb_dept1 中 name 列后添加一个 INT 类型的字段 column3，SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD column3 INT(11) AFTER name;
```

使用 DESC 查看表 tb_dept1，结果如下：

```
mysql> DESC tb_dept1;
+-----------+-----------------+----------+---------+------------+--------+
| Field     | Type            | Null     | Key     | Default    | Extra  |
+-----------+-----------------+----------+---------+------------+--------+
```

```
| column2  | int(11)      | YES    |      | NULL    |       |
| id       | int(11)      | NO     | PRI  | NULL    |       |
| name     | varchar(30)  | YES    |      | NULL    |       |
| column3  | int(11)      | YES    |      | NULL    |       |
| location    | varchar(60)  | YES    |      | NULL    |       |
| managerId | int(10)      | YES    |      | NULL    |       |
| column1  | varchar(12)    | NO     |      | NULL    |       |
+------------+----------------+-----------+---------+-------------+--------+
7 rows in set (0.03 sec)
```

可以看到，tb_dept1 表中增加了一个名称为 column3 的字段，其位置在指定的 name 字段后面，添加字段成功。

16.3.5 删除字段

删除字段是将数据表中的某个字段从表中移除，语法格式如下：

```
ALTER TABLE <表名> DROP <字段名>;
```

“字段名”指需要从表中删除的字段的名称。

【例 16.21】删除数据表 tb_dept1 表中的 column2 字段。

首先，执行删除字段之前，使用 DESC 查看 tb_dept1 表结构，结果如下：

```
mysql> DESC tb_dept1;
+-------------+---------------+----------+--------+-------------+--------+
| Field      | Type         | Null     | Key  | Default    | Extr  |
+-------------+---------------+----------+--------+-------------+--------+
| column2  | int(11)      | YES    |      | NULL    |       |
| id       | int(11)      | NO     | PRI  | NULL    |       |
| name     | varchar(30)  | YES    |      | NULL    |       |
| column3  | int(11)      | YES    |      | NULL    |       |
| location    | varchar(60)  | YES    |      | NULL    |       |
| managerId | int(10)      | YES    |      | NULL    |       |
| column1  | varchar(12)    | NO     |      | NULL    |       |
+--------------+-------------+-----------+---------+-------------+--------+
6 rows in set (0.03 sec)
```

删除 column2 字段，SQL 语句如下：

```
ALTER TABLE tb_dept1 DROP column2;
```

再次使用 DESC 查看表 tb_dept1，结果如下：

```
mysql> DESC tb_dept1;
+-------------+----------------+----------+---------+-------------+--------+
| Field      | Type         | Null     | Key  | Default    | Extr  |
+-------------+----------------+----------+---------+-------------+--------+
| id       | int(11)      | NO     | PRI  | NULL    |       |
```

```
| name      | varchar(30)  | YES   |      | NULL    |         |
| column3   | int(11)      | YES   |      | NULL    |         |
| location  | varchar(60)  | YES   |      | NULL    |         |
| managerId | int(10)      | YES   |      | NULL    |         |
| column1   | varchar(12)  | NO    |      | NULL    |         |
+-----------+--------------+-------+------+---------+---------+
6 rows in set (0.03 sec)
```

可以看到，tb_dept1 表中已经不存在名称为 column2 的字段，删除字段成功。

16.3.6 修改字段的排列位置

对于一个数据表来说，在创建的时候，字段在表中的排列顺序就已经确定了。但表的结构并不是完全不可以改变的，可以通过 ALTER TABLE 来改变表中字段的相对位置。语法格式如下：

```
ALTER TABLE <表名> MODIFY <字段1> <数据类型> FIRST|AFTER <字段2>;
```

“字段 1”指要修改位置的字段，“数据类型”指“字段 1”的数据类型，“FIRST”为可选参数，指将“字段 1”修改为表的第一个字段，“AFTER 字段 2”指将“字段 1”插入到“字段 2”后面。

1. 修改字段为表的第一个字段

【例 16.22】将数据表 tb_deptl 中的 column1 字段修改为表的第一个字段，SQL 语句如下：

```
ALTER TABLE tb_dept1 MODIFY column1 VARCHAR(12) FIRST;
```

使用 DESC 查看表 tb_dept1，发现字段 column1 已经被移至表的第一列，结果如下：

```
mysql> DESC tb_dept1;
+-----------+--------------+-------+------+---------+---------+
| Field     | Type         | Null  | Key  | Default | Extra   |
+-----------+--------------+-------+------+---------+---------+
| column1   | varchar(12)  | NO    |      | NULL    |         |
| id        | int(11)      | NO    | PRI  | NULL    |         |
| name      | varchar(30)  | YES   |      | NULL    |         |
| column3   | int(11)      | YES   |      | NULL    |         |
| location  | varchar(60)  | YES   |      | NULL    |         |
| managerId | int(10)      | YES   |      | NULL    |         |
+-----------+--------------+-------+------+---------+---------+
6 rows in set (0.03 sec)
```

2. 修改字段到表的指定列之后

【例 16.23】将数据表 tb_dept1 中的 column1 字段插入到 location 字段后面，SQL 语句如下：

```
ALTER TABLE tb_dept1 MODIFY column1 VARCHAR(12) AFTER location;
```

使用 DESC 查看表 tb_dept1，结果如下：

```
mysql> DESC tb_dept1;
```

```
+-------------+---------------+----------+--------+------------+--------+
| Field       | Type          | Null     | Key  | Default    | Extra  |
+-------------+---------------+----------+--------+------------+--------+
| id          | int(11)       | NO     | PRI    | NULL     |        |
| name      | varchar(30)  | YES     |        | NULL     |        |
| column3  | int(11)        | YES    |        | NULL     |        |
| location   | varchar(60)  | YES     |        | NULL     |        |
| column1  | varchar(12)   | NO     |        | NULL     |        |
| managerId | int(10)       | YES     |        | NULL     |        |
+-------------+---------------+----------+--------+------------+--------+
6 rows in set (0.03 sec)
```

可以看到，tb_dept1 表中的字段 column1 已经被移至 location 字段之后。

16.3.7　更改表的存储引擎

通过前面章节的学习，知道存储引擎是 MySQL 中的数据存储在文件或者内存中时采用的不同技术实现。可以根据自己的需要，选择不同的引擎，甚至可以为每一张表选择不同的存储引擎。MySQL 中主要存储引擎有：MyISAM、InnoDB、MEMORY（HEAP）、BDB、FEDERATED 等。可以使用 SHOW ENGINES;语句查看系统支持的存储引擎。表 16-3 列出了 MySQL 所支持的存储引擎。

表 16-3　MySQL 支持的存储引擎

引擎名	是否支持
FEDERATED	否
MRG_MYISAM	是
MyISAM	是
BLACKHOLE	是
CSV	是
MEMORY	是
ARCHIVE	是
InnoDB	默认
PERFORMANCE_SCHEMA	是

更改表的存储引擎的语法格式如下：

```
ALTER TABLE <表名> ENGINE=<更改后的存储引擎名>;
```

【例 16.24】将数据表 tb_deptment3 的存储引擎修改为 MyISAM。

在修改存储引擎之前，先使用 SHOW CREATE TABLE 查看表 tb_deptment3 当前的存储引擎，结果如下。

```
mysql>  SHOW CREATE TABLE tb_deptment3 \G
*** 1. row ***
```

```
      Table: tb_deptment3
Create Table: CREATE TABLE `tb_deptment3` (
  `id` int(11) NOT NULL,
  `name` varchar(22) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `STH` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

可以看到，表 tb_deptment3 当前的存储引擎为 ENGINE=InnoDB，接下来修改存储引擎类型，输入如下 SQL 语句并执行：

```
mysql> ALTER TABLE tb_deptment3 ENGINE=MyISAM;
```

使用 SHOW CREATE TABLE 再次查看表 tb_deptment3 的存储引擎，发现表 tb_dept 的存储引擎变成了“MyISAM”，结果如下：

```
mysql> SHOW CREATE TABLE tb_deptment3 \G
*** 1. row ***
     Table: tb_deptment3
Create Table: CREATE TABLE `tb_deptment3` (
  `id` int(11) NOT NULL,
  `name` varchar(22) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `STH` (`name`)
) ENGINE=MyISAM DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

16.3.8 删除表的外键约束

对于数据库中定义的外键，如果不再需要，可以将其删除。外键一旦删除，就会解除主表和从表间的关联关系，MySQL 中删除外键的语法格式如下：

```
ALTER TABLE <表名> DROP FOREIGN KEY <外键约束名>
```

“外键约束名”指在定义表时 CONSTRAINT 关键字后面的参数，详细内容可参考 16.1.3 节的“使用外键约束”。

【例 16.25】删除数据表 tb_emp9 中的外键约束。

首先创建表 tb_emp9，创建外键 deptId 关联 tb_dept1 表的主键 id，SQL 语句如下：

```
CREATE TABLE tb_emp9
(
id      INT(11) PRIMARY KEY,
name    VARCHAR(25),
```

```
deptId  INT(11),
salary   FLOAT,
CONSTRAINT fk_emp_dept  FOREIGN KEY (deptId) REFERENCES tb_dept1(id)
);
```

使用 SHOW CREATE TABLE 查看表 tb_emp9 的结构，结果如下：

```
mysql> SHOW CREATE TABLE tb_emp9 \G
*** 1. row ***
      Table: tb_emp9
Create Table: CREATE TABLE `tb_emp9` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`deptId`),
  CONSTRAINT `fk_emp_dept` FOREIGN KEY (`deptId`) REFERENCES `tb_dept1` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

可以看到，已经成功添加了表的外键，下面删除外键约束，SQL 语句如下：

```
ALTER TABLE tb_emp9 DROP FOREIGN KEY fk_emp_dept;
```

执行完毕之后，将删除表 tb_emp9 的外键约束，使用 SHOW CREATE TABLE 再次查看表 tb_emp9 结构，结果如下：

```
mysql> SHOW CREATE TABLE tb_emp9 \G
*** 1. row ***
      Table: tb_emp9
Create Table: CREATE TABLE `tb_emp9` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`deptId`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

可以看到，tb_emp9 中已经不存在 FOREIGN KEY，原有的名称为 fk_emp_dept 的外键约束删除成功。

16.4 删除数据表

删除数据表就是将数据库中已经存在的表从数据库中删除。注意，在删除表的同时，表的定义和表中所有的数据均会被删除。因此，在进行删除操作前，最好对表中的数据做个备份，以免造成无法挽回的后果。本节将详细讲解数据库表的删除方法。

16.4.1　删除没有被关联的表

在 MySQL 中，使用 DROP TABLE 可以一次删除一个或多个没有被其他表关联的数据表。语法格式如下：

```
DROP TABLE [IF EXISTS]表1, 表2,…表n;
```

其中“表 n”指要删除的表的名称，后面可以同时删除多个表，只需将要删除的表名依次写在后面，相互之间用逗号隔开即可。如果要删除的数据表不存在，则 MySQL 会提示一条错误信息，“ERROR 1051 (42S02): Unknown table '表名'”。参数“IF EXISTS”用于在删除前判断删除的表是否存在，加上该参数后，再删除表的时候，如果表不存在，SQL 语句可以顺利执行，但是会发出警告（warning）。

在前面的例子中，已经创建了名为 tb_dept2 的数据表。如果没有，读者可输入语句，创建该表，SQL 语句如例 16.7 所示。下面使用删除语句将该表删除。

【例 16.26】删除数据表 tb_dept2，SQL 语句如下：

```
DROP TABLE IF EXISTS tb_dept2;
```

语句执行完毕之后，使用 SHOW TABLES 命令查看当前数据库中所有的表，SQL 语句如下：

```
mysql> SHOW TABLES;
+---------------------+
| Tables_in_test_db |
+---------------------+
| tb_dept        |
| tb_deptment3   |
```

执行结果可以看到，数据表列表中已经不存在名称为 tb_dept2 的表，删除操作成功。

16.4.2　删除被其他表关联的主表

数据表之间存在外键关联的情况下，如果直接删除父表，结果会显示失败。原因是直接删除，将破坏表的参照完整性。如果必须要删除，可以先删除与它关联的子表，再删除父表，只是这样同时删除了两个表中的数据。但有的情况下可能要保留子表，这时如要单独删除父表，只需将关联的表的外键约束条件取消，然后就可以删除父表，下面讲解这种方法。

在数据库中创建两个关联表，首先，创建表 tb_dept2，SQL 语句如下：

```
CREATE TABLE tb_dept2
(
id       INT(11) PRIMARY KEY,
name     VARCHAR(22),
location  VARCHAR(50)
);
```

接下来创建表 tb_emp，SQL 语句如下：

```
CREATE TABLE tb_emp
(
id       INT(11) PRIMARY KEY,
name     VARCHAR(25),
```

```
    deptId   INT(11),
    salary   FLOAT,
    CONSTRAINT fk_emp_dept  FOREIGN KEY (deptId) REFERENCES tb_dept2(id)
    );
```

使用 SHOW CREATE TABLE 命令查看表 tb_emp 的外键约束，结果如下：

```
mysql> SHOW CREATE TABLE tb_emp\G
*** 1. row ***
      Table: tb_emp
Create Table: CREATE TABLE `tb_emp` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`deptId`),
  CONSTRAINT `fk_emp_dept` FOREIGN KEY (`deptId`) REFERENCES `tb_dept2` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

可以看到，以上执行结果创建了两个关联表 tb_dept2 和表 tb_emp，其中 tb_emp 表为子表，具有名称为 fk_emp_dept 的外键约束，tb_dept2 为父表，其主键 id 被子表 tb_emp 所关联。

【例 16.27】删除被数据表 tb_emp 关联的数据表 tb_dept2。

首先直接删除父表 tb_dept2，输入删除语句如下：

```
mysql> DROP TABLE tb_dept2;
ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key constraint fails
```

可以看到，如前所述，在存在外键约束时，主表不能被直接删除。

接下来，解除关联子表 tb_emp 的外键约束，SQL 语句如下：

```
ALTER TABLE tb_emp DROP FOREIGN KEY fk_emp_dept;
```

语句成功执行后，将取消表 tb_emp 和表 tb_dept2 之间的关联关系，此时，可以输入删除语句，将原来的父表 tb_dept2 删除，SQL 语句如下：

```
DROP TABLE tb_dept2;
```

最后通过 SHOW TABLES;查看数据表列表，如下所示：

```
mysql> show tables;
+---------------------+
| Tables_in_test_db |
+---------------------+
| tb_dept          |
| tb_deptment3     |
......省略部分内容
```

可以看到，数据表列表中已经不存在名称为 tb_dept2 的表。

16.5 实战演练——数据表的基本操作

本章全面介绍了 MySQL 中数据表的各种操作，如创建表、添加各类约束、查看表结构，以及修改和删除表。读者应该掌握这些基本的操作，为以后的学习打下坚实的基础。在这里，给出一个综合案例，让读者全面回顾一下本章的知识要点，并通过这些操作来检验自己是否已经掌握了数据表的常用操作。

1. 案例目的

创建、修改和删除表，掌握数据表的基本操作。

创建数据库 company，按照表 16-4 和表 16-5 给出的表结构在 company 数据库中创建两个数据表 offices 和 employees，按照操作过程完成对数据表的基本操作。

表 16-4 offices 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
officeCode	INT(10)	是	否	是	是	否
city	VARCHAR(50)	否	否	是	否	否
address	VARCHAR(50)	否	否	否	否	否
country	VARCHAR(50)	否	否	是	否	否
postalCode	VARCHAR(15)	否	否	否	是	否

表 16-5 employees 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
employeeNumber	INT(11)	是	否	是	是	是
lastName	VARCHAR(50)	否	否	是	否	否
firstName	VARCHAR(50)	否	否	是	否	否
mobile	VARCHAR(25)	否	否	否	是	否
officeCode	INT(10)	否	是	是	否	否
jobTitle	VARCHAR(50)	否	否	是	否	否
birth	DATETIME	否	否	是	否	否
note	VARCHAR(255)	否	否	否	否	否
sex	VARCHAR(5)	否	否	否	否	否

2. 案例操作过程

01 登录 MySQL 数据库。

打开 windows 命令行，输入登录用户名和密码：

```
C:\>mysql -h localhost -u root -p
Enter password: **
```

或者打开 MySQL 5.7 Command Line Client，只用输入用户密码也可以登录。登录成功后显示

如下信息：

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.10 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

登录成功，可以输入 SQL 语句进行操作。

02 创建数据库 company。

创建数据库 company 的语句如下：

```
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.00 sec)
```

结果显示创建成功，在 company 数据库中创建表，必须先选择该数据库，输入语句如下：

```
mysql> USE company;
Database changed
```

结果显示选择数据库成功。

03 创建表 offices。

创建表 offices 的语句如下：

```
CREATE TABLE offices
(
officeCode  INT(10) NOT NULL UNIQUE,
city       VARCHAR(50) NOT NULL,
address     VARCHAR(50) NOT NULL,
country     VARCHAR(50) NOT NULL,
postalCode  VARCHAR(15) NOT NULL,
PRIMARY KEY  (officeCode)
);
```

执行成功之后，使用 SHOW TABLES;语句查看数据库中的表，语句如下：

```
mysql> show tables;
+-----------------------+
```

```
| Tables_in_company |
+-----------------------+
| offices           |
+-----------------------+
1 row in set (0.00 sec)
```

可以看到，数据库中已经有了数据表 offices，创建成功。

04 创建表 employees。

创建表 employees 的语句如下：

```
CREATE TABLE employees
(
employeeNumber  INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
lastName        VARCHAR(50) NOT NULL,
firstName       VARCHAR(50) NOT NULL,
mobile          VARCHAR(25) NOT NULL,
officeCode      INT(10) NOT NULL,
jobTitle        VARCHAR(50) NOT NULL,
birth           DATETIME,
note            VARCHAR(255),
sex             VARCHAR(5),
CONSTRAINT office_fk FOREIGN KEY(officeCode)  REFERENCES offices(officeCode)
);
```

执行成功之后，使用 SHOW TABLES;语句查看数据库中的表，语句如下：

```
mysql> show tables;
+-----------------------+
| Tables_in_company |
+-----------------------+
| employees          |
| offices            |
+-----------------------+
2 rows in set (0.00 sec)
```

可以看到，现在数据库中已经创建好了 employees 和 offices 两个数据表。要检查表的结构是否按照要求创建，可使用 DESC 分别查看两个表的结构，如果语句正确，则显示结果如下：

```
mysql>DESC offices;
+--------------+-------------+-------+------+---------+-------+
| Field       | Type       | Null  | Key | Default | Extra |
+--------------+-------------+-------+------+---------+-------+
| officeCode  | int(10)    | NO    | PRI | NULL    |       |
| city        | varchar(50) | NO   |     | NULL    |       |
| address     | varchar(50) | NO   |     | NULL    |       |
| country     | varchar(50) | NO   |     | NULL    |       |
| postalCode  | varchar(15) | NO   |     | NULL    |       |
```

```
+------------+---------------+--------+-----+---------+-------+
5 rows in set (0.02 sec)

mysql>DESC employees;
+----------------------+--------------+--------+-------+---------+---------
-----------+
| Field            | Type       | Null  | Key | Default | Extra          |
+----------------------+--------------+--------+-------+---------+---------
-----------+
| employeeNumber | int(11)      | NO   | PRI  | NULL | auto_increment |
| lastName        | varchar(50)  | NO   |      | NULL |                |
| firstName       | varchar(50)  | NO   |      | NULL |                |
| mobile          | varchar(25)  | NO   |      | NULL |                |
| officeCode      | int(10)      | NO   | MUL  | NULL |                |
| jobTitle        | varchar(50)  | NO   |      | NULL |                |
| birth           | datetime     | YES  |      | NULL |                |
| note            | varchar(255) | YES  |      | NULL |                |
| sex             | varchar(5)   | YES  |      | NULL |                |
+--------------------+-----------------+--------+-------+---------+---------
-----------+
9 rows in set (0.00 sec)
```

可以看到，两个表中字段分别满足表 16-2 和表 16-3 中要求的数据类型和约束类型。

05 将表 employees 的 mobile 字段修改到 officeCode 字段后面。

修改字段位置，需要用到 ALTER TABLE 语句，输入语句如下：

```
mysql> ALTER TABLE employees MODIFY mobile VARCHAR(25) AFTER officeCode;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

结果显示执行成功，使用 DESC 查看修改后的结果如下：

```
mysql>DESC employees;
+----------+-------------+--------+-------+---------+---------------+
| Field            | Type       | Null  | Key | Default | Extra          |
+----------+-------------+--------+-------+---------+---------------+
| employeeNumber | int(11)      | NO   | PRI  | NULL | auto_increment |
| lastName        | varchar(50)  | NO   |      | NULL |                |
| firstName       | varchar(50)  | NO   |      | NULL |                |
| officeCode      | int(10)      | NO   | MUL  | NULL |                |
| mobile         | varchar(25) | NO   |      | NULL |               |
| jobTitle        | varchar(50)  | NO   |      | NULL |                |
| employee _birth | datetime      | YES  |      | NULL |                |
| note            | varchar(255) | YES  |      | NULL |                |
| sex             | varchar(5)  | YES  |      | NULL |               |
+--------+---------------+--------+-------+---------+---------------+
```

```
9 rows in set (0.00 sec)
```

可以看到，mobile 字段已经插入到 officeCode 字段的后面。

06 将表 employees 的 birth 字段改名为 employee_birth。

修改字段名，需要用到 ALTER TABLE 语句，输入语句如下：

```
ALTER TABLE employees CHANGE birth employee_birth DATETIME;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

结果显示执行成功，使用 DESC 查看修改后的结果如下：

```
mysql>DESC employees;
+----------------------+-------------+--------+-------+---------+---------
-----------+
| Field          | Type       | Null  | Key | Default | Extra          |
+--------------+--------------+--------+-------+---------+----------------+
| employeeNumber | int(11)      | NO    | PRI  | NULL | auto_increment |
| lastName       | varchar(50)  | NO    |      | NULL |                |
| firstName      | varchar(50)  | NO    |      | NULL |                |
| mobile         | varchar(25)  | NO    |      | NULL |                |
| officeCode     | int(10)      | NO    | MUL  | NULL |                |
| jobTitle       | varchar(50)  | NO    |      | NULL |                |
| employee _birth | datetime    | YES   |      | NULL |                |
| note           | varchar(255) | YES   |      | NULL |                |
| sex            | varchar(5)   | YES   |      | NULL |                |
+-------------+---------------+--------+-------+---------+--------------+
9 rows in set (0.00 sec)
```

可以看到，表中只有 employee_birth 字段，已经没有名称为 birth 的字段了，修改名称成功。

07 修改 sex 字段，数据类型为 CHAR(1)，非空约束。

修改字段数据类型，需要用到 ALTER TABLE 语句，输入语句如下：

```
mysql>ALTER TABLE employees MODIFY sex CHAR(1) NOT NULL;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

结果显示执行成功，使用 DESC 查看修改后的结果如下：

```
mysql>DESC employees;
+--------------+--------------+--------+-------+---------+------------+
| Field          | Type       | Null  | Key | Default | Extra          |
+--------------+--------------+--------+-------+---------+------------+
| employeeNumber | int(11)      | NO    | PRI  | NULL | auto_increment |
| lastName       | varchar(50)  | NO    |      | NULL |                |
| firstName      | varchar(50)  | NO    |      | NULL |                |
```

```
| mobile         | varchar(25)  | NO   |     | NULL |               |
| officeCode     | int(10)      | NO   | MUL | NULL |               |
| jobTitle       | varchar(50)  | NO   |     | NULL |               |
| employee _birth | datetime    | YES  |     | NULL |               |
| note           | varchar(255) | YES  |     | NULL |               |
| sex            | char(1)      | NO   |     | NULL |               |
+------------+---------------+--------+-------+---------+------------+
9 rows in set (0.00 sec)
```

执行结果可以看到，sex 字段的数据类型由前面的 VARCHAR(5)修改为 CHAR(1)，且其 Null 列显示为 NO，表示该列不允许空值，修改成功。

08 删除字段 note。

删除字段，需要用到 ALTER TABLE 语句，输入语句如下：

```
mysql> ALTER TABLE employees DROP note;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

结果显示执行语句成功，使用 DESC employees;查看语句执行后的结果：

```
mysql> desc employees;
+--------------+--------------+--------+-------+---------+------------+
| Field          | Type        | Null  | Key | Default | Extra          |
+--------------+--------------+--------+-------+---------+------------+
|employeeNumber | int(11)      | NO    | PRI  | NULL | auto_increment |
| lastName       | varchar(50) | NO    |      | NULL |               |
| firstName      | varchar(50) | NO    |      | NULL |               |
| mobile         | varchar(25) | NO    |      | NULL |               |
| officeCode     | int(10)     | NO    | MUL  | NULL |               |
| jobTitle       | varchar(50) | NO    |      | NULL |               |
| employee _birth | datetime   | YES   |      | NULL |               |
| sex            | char(1)     | NO    |      | NULL |               |
+------------+---------------+--------+-------+---------+------------+
8 rows in set (0.00 sec)
```

可以看到，DESC 语句返回了 8 个列字段，note 字段已经不在表结构中，删除字段成功。

09 增加字段名 favoriate_activity，数据类型为 VARCHAR(100)。

增加字段，需要用到 ALTER TABLE 语句，输入语句如下：

```
mysql> ALTER TABLE employees ADD favoriate_activity VARCHAR(100);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

结果显示执行语句成功，使用 DESC employees;查看语句执行后的结果：

```
mysql> desc employees;
```

```
+--------------+--------------+--------+-------+---------+------------+
| Field          | Type       | Null  | Key | Default | Extra        |
+--------------+--------------+--------+-------+---------+------------+
|employeeNumber | int(11)      | NO    | PRI  | NULL | auto_increment |
| lastName        | varchar(50)  | NO   |      | NULL |              |
| firstName        | varchar(50)  | NO   |      | NULL |              |
| mobile          | varchar(25)  | NO   |      | NULL |              |
| officeCode       | int(10)       | NO   | MUL | NULL |              |
| jobTitle         | varchar(50)  | NO   |      | NULL |              |
| employee _birth  | datetime     | YES  |      | NULL |              |
| sex             | char(1)      | NO  |      | NULL |              |
| favoriate_activity | varchar(100) | YES  |      | NULL |              |
+--------------+----------------+--------+-------+---------+------------+
9 rows in set (0.00 sec)
```

可以看到，数据表 employees 中增加了一个新的列 favoriate_activity，数据类型为 VARCHAR(100)，允许空值，添加新字段成功。

10 删除表 offices。

在创建表 employees 时，设置了表的外键，该表关联了其父表的 officeCode 主键。如前面所述，删除关联表时，要先删除子表 employees 的外键约束，才能删除父表。因此，必须先删除 employees 表的外键约束。

（1）删除 employees 表的外键约束，输入如下语句：

```
mysql>ALTER TABLE employees DROP FOREIGN KEY office_fk;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

其中 office_fk 为 employees 表的外键约束的名称，即创建外键约束时 CONSTRAINT 关键字后面的参数，结果显示语句执行成功，现在可以删除 offices 父表。

（2）删除表 offices，输入如下语句：

```
mysql>DROP TABLE offices;
Query OK, 0 rows affected (0.00 sec)
```

结果显示执行删除操作成功，使用 SHOW TABLES;语句查看数据库中的表，结果如下：

```
mysql> show tables;
+-----------------------+
| Tables_in_company |
+-----------------------+
| employees         |
+-----------------------+
1 row in set (0.00 sec)
```

可以看到，数据库中已经没有名称为 offices 的表了，删除表成功。

11 修改表 employees 存储引擎为 MyISAM。

修改表存储引擎，需要用到 ALTER TABLE 语句，输入语句如下：

```
mysql>ALTER TABLE employees ENGINE=MyISAM;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

结果显示执行修改存储引擎操作成功，使用 SHOW CREATE TABLE 语句查看表结构，结果如下：

```
mysql> show CREATE TABLE employees\G
*** 1. row ***
      Table: employees
Create Table: CREATE TABLE `employees` (
  `employeeNumber` int(11) NOT NULL AUTO_INCREMENT,
  `lastName` varchar(50) NOT NULL,
  `firstName` varchar(50) NOT NULL,
  `officeCode` int(10) NOT NULL,
  `mobile` varchar(25) DEFAULT NULL,
  `jobTitle` varchar(50) NOT NULL,
  `employee_birth` datetime DEFAULT NULL,
  `sex` char(1) NOT NULL,
  `favoriate_activity` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`employeeNumber`),
  KEY `office_fk` (`officeCode`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

可以看到，倒数第 2 行中的 ENGINE 后面的参数已经修改为 MyISAM，修改成功。

12 将表 employees 名称修改为 employees_info。

修改数据表名，需要用到 ALTER TABLE 语句，输入语句如下：

```
mysql>ALTER TABLE employees RENAME employees_info;
Query OK, 0 rows affected (0.00 sec)
```

结果显示执行语句成功，使用 SHOW TABLES;语句查看执行结果：

```
mysql> show tables;
+-----------------------+
| Tables_in_company |
+-----------------------+
| employees_info   |
+----------------------+
1 rows in set (0.00 sec)
```

可以看到数据库中已经没有名称为 employees 的数据表。

16.6　高手私房菜

技巧 1：表删除操作须谨慎。

表删除操作将把表的定义和表中的数据一起删除，并且 MySQL 在执行删除操作时，不会有任何的确认信息提示，因此执行删除操时，应当慎重。在删除表前，最好对表中的数据进行备份，这样当操作失误时，可以对数据进行恢复，以免造成无法挽回的后果。

同样的，在使用 ALTER TABLE 进行表的基本修改操作时，在执行操作过程之前，也应该确保对数据进行完整的备份，因为数据库的改变是无法撤销的，如果添加了一个不需要的字段，可以将其删除；相同的，如果删除了一个需要的列，该列下面的所有数据都将会丢失。

技巧 2：每一个表中都要有一个主键吗？

并不是每一个表中都需要主键，一般的，如果多个表之间进行连接操作时，需要用到主键。因此并不需要为每个表建立主键，而且有些情况最好不使用主键。

技巧 3：并不是每个表都可以任意选择存储引擎。

外键约束（FOREIGN KEY）不能跨引擎使用。MySQL 支持多种存储引擎，每一个表都可以指定一个不同的存储引擎，但是要注意：外键约束是用来保证数据的参照完整性，如果表之间需要关联外键，却指定了不同的存储引擎，这些表之间是不能创建外键约束的。所以说，存储引擎的选择也不完全是随意的。

技巧 4：带 AUTO_INCREMENT 约束的字段值是从 1 开始的吗？

默认的，在 MySQL 中，AUTO_INCREMENT 的初始值是 1，每新增一条记录，字段值自动加 1。设置自增属性（AUTO_INCREMENT）的时候，还可以指定第一条插入记录的自增字段的值，这样新插入的记录的自增字段值从初始值开始递增，如在 tb_emp8 中插入第一条记录，同时指定 id 值为 5，则以后插入的记录的 id 值就会从 6 开始往上增加。添加唯一性的主键约束时，往往需要设置字段自动增加属性。

16.7　经典习题

1. 创建数据库 Market，在 Market 中创建数据表 customers，customers 表结构如表 16-6 所示，按要求进行操作。

表 16-6　customers 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
c_num	INT(11)	是	否	是	是	是
c_name	VARCHAR(50)	否	否	否	否	否
c_contact	VARCHAR(50)	否	否	否	否	否
c_city	VARCHAR(50)	否	否	否	否	否
c_birth	DATETIME	否	否	是	否	否

（1）创建数据库 Market。

（2）创建数据表 customers，在 c_num 字段上添加主键约束和自增约束，在 c_birth 字段上添加非空约束。

（3）将 c_contact 字段插入到 c_birth 字段后面。

（4）将 c_name 字段数据类型改为 VARCHAR(70)。

（5）将 c_contact 字段改名为 c_phone。

（6）增加 c_gender 字段，数据类型为 CHAR(1)。

（7）将表名修改为 customers_info。

（8）删除字段 c_city。

（9）修改数据表的存储引擎为 MyISAM。

2. 在 Market 中创建数据表 orders，orders 表结构如表 16-7 所示，按要求进行操作。

表 16-7　orders 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
o_num	INT(11)	是	否	是	是	是
o_date	DATE	否	否	否	否	否
c_id	VARCHAR(50)	否	是	否	否	否

（1）创建数据表 orders，在 o_num 字段上添加主键约束和自增约束，在 c_id 字段上添加外键约束，关联 customers 表中的主键 c_num。

（2）删除 orders 表的外键约束，然后删除表 customers。

第 17 章　数据的基本操作

数据库管理系统的一个最重要的功能就是提供数据的各种操作，包括插入数据、更新数据、删除数据和查询数据等。其中数据查询不是简单返回数据库中存储的数据，而是应该根据需要对数据进行筛选，以及数据将以什么样的格式显示。MySQL 提供了功能强大、灵活的语句来实现这些操作。本章将介绍数据的这些基本操作方法和技巧。

本章学习目标

- 掌握如何插入数据
- 掌握如何更新数据
- 掌握如何删除数据
- 掌握如何查询数据
- 熟练综合案例数据的基本操作

17.1　插入数据

在使用数据库之前，数据库中必须要有数据，MySQL 中使用 INSERT 语句向数据表中插入新的数据记录。可以插入的方式有：插入完整的记录，插入记录的一部分，插入多条记录，以及插入另一个查询的结果。下面将介绍这些内容。

17.1.1　为表的所有字段插入数据

使用基本的 INSERT 语句插入数据要求指定表的名称和插入到新记录中的值。基本语法格式为：

```
INSERT INTO table_name (column_list) VALUES (value_list);
```

table_name 指定要插入数据的表名，column_list 指定要插入数据的那些列，value_list 指定每个列对应插入的数据。注意，使用该语句时字段列和数据值的数量必须相同。

本章将使用样例表 person，创建语句如下：

```
CREATE TABLE person
(
    id     INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name   CHAR(40) NOT NULL DEFAULT '',
    age    INT NOT NULL DEFAULT 0,
    info   CHAR(50) NULL,
    PRIMARY KEY (id)
);
```

向表中所有字段插入值的方法有两种：一种是指定所有字段名，另一种是完全不指定字段名。

【例 17.1】在表 person 中，插入一条新记录，id 值为 1，name 值为 Green，age 值为 21，info 值为 lawyer，SQL 语句如下：

执行插入操作之前，使用 SELECT 语句查看表中的数据：

```
mysql> SELECT * FROM person;
Empty set (0.00 sec)
```

结果显示当前表为空，没有数据，接下来执行插入操作：

```
mysql> INSERT INTO person (id ,name, age , info)
    -> VALUES (1,'Green', 21, 'Lawyer');
Query OK, 1 row affected (0.00 sec)
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person;
+----+--------+-----+------------+
| id | name   | age | info       |
+----+--------+-----+------------+
|  1 | Green  |  21 | Lawyer     |
+----+--------+-----+------------+
```

可以看到插入记录成功。在插入数据时，指定了表 person 的所有字段，因此将为每一个字段插入新的值。

INSERT 语句后面的列名称顺序可以不是表 person 定义时的顺序。即插入数据时，不需要按照表定义的顺序插入，只要保证值的顺序与列字段的顺序相同就可以，如下面的例子。

【例 17.2】在表 person 中，插入一条新记录，id 值为 2，name 值为 Suse，age 值为 22，info 值为 dancer，SQL 语句如下：

```
mysql> INSERT INTO person (age ,name, id , info)
    -> VALUES (22, 'Suse', 2, 'dancer');
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person;
+----+--------+-----+------------+
| id | name   | age | info       |
+----+--------+-----+------------+
|  1 | Green  |  21 | Lawyer     |
|  2 | Suse   |  22 | dancer     |
+----+--------+-----+------------+
```

由结果可以看到，INSERT 语句成功插入了一条记录。

使用 INSERT 插入数据时，允许列名称列表 column_list 为空，此时，值列表中需要为表的每一个字段指定值，并且值的顺序必须和数据表中字段定义时的顺序相同。

【例 17.3】在表 person 中，插入一条新记录，SQL 语句如下：

```
mysql> INSERT INTO person
    -> VALUES (3,'Mary', 24, 'Musician');
Query OK, 1 row affected (0.00 sec)
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person;
+----+--------+-----+------------+
| id | name   | age | info       |
+----+--------+-----+------------+
|  1 | Green  |  21 | Lawyer     |
|  2 | Suse   |  22 | dancer     |
|  3 | Mary   |  24 | Musician   |
+----+--------+-----+------------+
```

可以看到插入记录成功。数据库中增加了一条 id 为 3 的记录，其他字段值为指定的插入值。本例的 INSERT 语句中没有指定插入列表，只有一个值列表。在这种情况下，值列表为每一个字段列指定插入值，并且这些值的顺序必须和表 person 中字段定义的顺序相同。


提 示

虽然使用 INSERT 插入数据时可以忽略插入数据的列名称，但是如果不包含列名称，那么 VALUES 关键字后面的值不仅要求必须完整而且顺序必须和表定义时列的顺序相同。如果表的结构被修改，则对列进行增加、删除或者位置改变操作，将使得用这种方式插入数据时的顺序也必须同时改变。如果指定列名称，则不会受到表结构改变的影响。

17.1.2　为表的指定字段插入数据

为表的指定字段插入数据，就是在 INSERT 语句中只向部分字段中插入值，而其他字段的值为表定义时的默认值。

【例 17.4】在表 person 中，插入一条新记录，name 值为 Willam，age 值为 20，info 值为 sports man，SQL 语句如下：

```
mysql> INSERT INTO person (name, age,info)
    -> VALUES('Willam', 20, 'sports man');
Query OK, 1 row affected (0.00 sec)
```

提示信息表示插入一条记录成功。使用 SELECT 查询表中的记录，查询结果如下：

```
mysql> SELECT * FROM person;
+----+--------+-----+------------+
```

```
| id | name    | age | info        |
+----+---------+-----+-------------+
|  1 | Green   |  21 | Lawyer      |
|  2 | Suse    |  22 | dancer      |
|  3 | Mary    |  24 | Musician    |
|  4 | Willam |  20  | sports man |
+----+---------+-----+-------------+
```

可以看到插入记录成功。如这里的 id 字段，查询结果显示，该字段自动添加了一个整数值 4。在这里 id 字段为表的主键，不能为空，系统会自动为该字段插入自增的序列值。在插入记录时，如果某些字段没有指定插入值，MySQL 将插入该字段定义时的默认值。下面例子说明在没有指定列字段时，插入默认值。

【例 17.5】在表 person 中，插入一条新记录，name 值为 Laura，age 值为 25，SQL 语句如下：

```
mysql> INSERT INTO person (name, age ) VALUES ('Laura', 25);
Query OK, 1 row affected (0.00 sec)
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person;
+----+---------+-----+-------------+
| id | name    | age | info        |
+----+---------+-----+-------------+
|  1 | Green   |  21 | Lawyer      |
|  2 | Suse    |  22 | dancer      |
|  3 | Mary    |  24 | Musician    |
|  4 | Willam |  20 | sports man |
|  5 | Laura   |  25 | NULL        |
+----+---------+-----+-------------+
```

可以看到，在本例插入语句中，没有指定 info 字段值，查询结果显示，info 字段在定义时指定默认值为 NULL，因此系统自动为该字段插入空值。



提 示

要保证每个插入值的类型必须和对应列的数据类型匹配，如果类型不同，将无法插入，并且 MySQL 会产生错误。

17.1.3 同时插入多条记录

INSERT 语句可以同时向数据表中插入多条记录，插入时指定多个值列表，每个值列表之间用逗号分隔开，基本语法格式如下：

```
INSERT INTO table_name (column_list)
VALUES (value_list1),  (value_list2),...,(value_listn);
```

“value_list1，value_list2，…value_listn；”分别表示第 n 个插入记录的字段的值列表。

【例 17.6】在表 person 中，在 name、age 和 info 字段指定插入值，同时插入 3 条新记录，SQL 语句如下：

```
INSERT INTO person(name, age, info)
VALUES ('Evans',27, 'secretary'),
       ('Dale',22, 'cook'),
       ('Edison',28, 'singer');
```

语句执行结果如下：

```
mysql> INSERT INTO person(name, age, info)
    -> VALUES ('Evans',27, 'secretary'),
    -> ('Dale',22, 'cook'),
    -> ('Edison',28, 'singer');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

语句执行完毕，查看执行结果：

```
mysql>  SELECT * FROM person;
+----+--------+-----+------------+
| id | name   | age | info       |
+----+--------+-----+------------+
|  1 | Green  |  21 | Lawyer     |
|  2 | Suse   |  22 | dancer     |
|  3 | Mary   |  24 | Musician   |
|  4 | Willam |  20  | sports man  |
|  5 | Laura  |  25 | NULL       |
|  6 | Evans  |  27 | secretary  |
|  7 | Dale   |  22 | cook       |
|  8 | Edison |  28  | singer      |
+----+--------+-----+------------+
```

由结果可以看到，INSERT 语句执行后，表 person 中添加了 3 条记录，其 name 和 age 字段分别为指定的值，id 字段为 MySQL 添加的默认的自增值。

使用 INSERT 同时插入多条记录时，MySQL 会返回一些在执行单行插入时没有的额外信息，这些包含数值的字符串的意思分别如下。

- Records：表明插入的记录条数。
- Duplicates：表明插入时被忽略的记录，原因可能是这些记录包含了重复的主键值。
- Warnings：表明有问题的数据值，例如发生数据类型转换。

【例 17.7】在表 person 中，不指定插入列表，同时插入两条新记录，SQL 语句如下：

```
INSERT INTO person
```

```
VALUES (9,'Harry',21, 'magician'),
       (NULL,'Harriet',19, 'pianist');
```

语句执行结果如下：

```
mysql> INSERT INTO person
    -> VALUES (9,'Harry',21, 'magician'),
    -> (NULL,'Harriet',19, 'pianist');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person;
+----+---------+-----+------------+
| id | name    | age | info       |
+----+---------+-----+------------+
|  1 | Green   |  21 | Lawyer     |
|  2 | Suse    |  22 | dancer     |
|  3 | Mary    |  24 | Musician   |
|  4 | Willam  |  20 | sports man |
|  5 | Laura   |  25 | NULL       |
|  6 | Evans   |  27 | secretary  |
|  7 | Dale    |  22 | cook       |
|  8 | Edison  |  28 | singer     |
|  9 | Harry   |  21 | magician   |
| 10 | Harriet |  19 | pianist    |
+----+---------+-----+------------+
```

由结果可以看到，INSERT 语句执行后，表 person 中添加了两条记录，与前面介绍单个 INSERT 语法不同，person 表名后面没有指定插入字段列表，因此 VALUES 关键字后面的多个值列表都要为每一条记录的每一个字段列指定插入值，并且这些值的顺序必须和表 person 中字段定义的顺序相同，带有 AUTO_INCREMENT 属性的 id 字段插入 NULL 值，系统会自动为该字段插入唯一的自增编号。



提 示

一个同时插入多行记录的 INSERT 语句可以等同于多个单行插入的 INSERT 语句，但是同时插入多行的 INSERT 语句在处理过程中效率更高。所以在插入多条记录时，最好选择使用单条 INSERT 语句的方式插入。

17.2 更新数据

表中有数据之后，接下来可以对数据进行更新操作，MySQL 中使用 UPDATE 语句更新表中的记录，可以更新特定的行或者同时更新所有的行。基本语法结构如下：

```
UPDATE table_name
SET column_name1 = value1,column_name2=value2,…,column_namen=valuen
WHERE (condition);
```

“column_name1,column_name2,…,column_namen”为指定更新的字段的名称；“value1, value2,…valuen”为相对应的指定字段的更新值；condition 指定更新的记录需要满足的条件。更新多个列时，每个“列-值”对之间用逗号隔开，最后一列之后不需要逗号。

【例 17.8】在表 person 中，更新 id 值为 10 的记录，将 age 字段值改为 15，将 name 字段值改为 LiMing，SQL 语句如下：

```
UPDATE person SET age = 15, name= 'LiMing' WHERE id = 10;
```

更新操作执行前可以使用 SELECT 语句查看当前的数据：

```
mysql> SELECT * FROM person WHERE id=10;
+----+-------+-----+---------+
| id | name  | age | info    |
+----+-------+-----+---------+
| 10 | Harry |  20 | student |
+----+-------+-----+---------+
```

由结果可以看到更新之前，id 等于 10 的记录的 name 字段值为 harry，age 字段值为 20，下面使用 UPDATE 语句更新数据，语句执行结果如下：

```
mysql> UPDATE person SET age = 15, name='LiMing' WHERE id = 10;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person WHERE id=10;
+----+--------+-----+---------+
| id | name   | age | info    |
+----+--------+-----+---------+
| 10 | LiMing |  15 | student |
+----+--------+-----+---------+
```

由结果可以看到，id 等于 10 的记录中的 name 和 age 字段的值已经成功被修改为指定值。


提 示

保证 UPDATE 以 WHERE 子句结束，通过 WHERE 子句指定被更新的记录所需要满足的条件，如果忽略 WHERE 子句，MySQL 将更新表中所有的行。

【例 17.9】在表 person 中，更新 age 值为 19 到 22 的记录，将 info 字段值都改为 student，SQL 语句如下：

```
UPDATE person SET info= 'student' WHERE id BETWEEN 19 AND 22;
```

更新操作执行前可以使用 SELECT 语句查看当前的数据：

```
mysql> SELECT * FROM person WHERE age BETWEEN 19 AND 22;
+----+---------+-----+------------+
| id | name    | age | info       |
+----+---------+-----+------------+
|  1 | Willam  |  20 | sports man |
|  3 | Green   |  21 | Lawyer     |
|  4 | Suse    |  22 | dancer     |
|  6 | Dale    |  22 | cook       |
|  8 | Harry   |  21 | magician   |
|  9 | Harriet |  19 | pianist    |
+----+---------+-----+------------+
```

可以看到，这些 age 字段值在 19 到 22 之间的记录的 info 字段值各不相同。下面使用 UPDATE 语句更新数据，语句执行结果如下：

```
mysql> UPDATE person SET info='student' WHERE age BETWEEN 19 AND 22;
Query OK, 6 rows affected (0.00 sec)
Rows matched: 6  Changed: 6  Warnings: 0
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person WHERE age BETWEEN 19 AND 22;
+----+---------+-----+---------+
| id | name    | age | info    |
+----+---------+-----+---------+
|  1 | Willam  |  20 | student |
|  3 | Green   |  21 | student |
|  4 | Suse    |  22 | student |
|  6 | Dale    |  22 | student |
|  8 | Harry   |  21 | student |
|  9 | Harriet |  19 | student |
+----+---------+-----+---------+
```

由结果可以看到，UPDATE 执行后，成功地将表中符合条件的记录的 info 字段值都改为 student。

17.3 删除数据

从数据表中删除数据使用 DELETE 语句，DELETE 语句允许 WHERE 子句指定删除条件。DELETE 语句基本语法格式如下：

```
DELETE FROM table_name [WHERE <condition>];
```

table_name 指定要执行删除操作的表；“[WHERE <condition>]”为可选参数，指定删除条件，如果没有 WHERE 子句，DELETE 语句将删除表中的所有记录。

【例 17.10】在表 person 中，删除 id 等于 10 的记录，SQL 语句如下：

执行删除操作前使用 SELECT 语句查看当前 id=10 的记录：

```
mysql> SELECT * FROM person WHERE id=10;
+----+--------+-----+---------+
| id | name   | age | info    |
+----+--------+-----+---------+
| 10 | LiMing |  15 | student |
+----+--------+-----+---------+
```

可以看到，现在表中有 id=10 的记录，下面使用 DELETE 语句删除记录，语句执行结果如下：

```
mysql> DELETE FROM person WHERE id = 10;
Query OK, 1 row affected (0.02 sec)
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person WHERE id=10;
Empty set (0.00 sec)
```

查询结果为空，说明删除操作成功。

【例 17.11】在 person 表中，使用 DELETE 语句同时删除多条记录，在前面 UPDATE 语句中将 age 字段值在 19 到 22 之间的记录的 info 字段值修改为 student，在这里删除这些记录，SQL 语句如下：

```
DELETE FROM person WHERE age BETWEEN 19 AND 22;
```

执行删除操作前使用 SELECT 语句查看当前的数据：

```
mysql> SELECT * FROM person WHERE age BETWEEN 19 AND 22;
+----+---------+-----+---------+
| id | name    | age | info    |
+----+---------+-----+---------+
|  1 | Willam  |  20 | student |
|  3 | Green   |  21 | student |
|  4 | Suse    |  22 | student |
|  6 | Dale    |  22 | student |
|  8 | Harry   |  21 | student |
|  9 | Harriet |  19 | student |
+----+---------+-----+---------+
```

可以看到，这些 age 字段值在 19 到 22 之间的记录存在表中。下面使用 DELETE 删除这些记录：

```
mysql> DELETE FROM person WHERE age BETWEEN 19 AND 22;
```

```
Query OK, 6 rows affected (0.00 sec)
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person WHERE age BETWEEN 19 AND 22;
Empty set (0.00 sec)
```

查询结果为空，删除多条记录成功。

【例 17.12】删除 person 表中所有记录，SQL 语句如下：

```
DELETE FROM person;
```

执行删除操作前使用 SELECT 语句查看当前的数据：

```
mysql> SELECT * FROM person;
+----+---------+-----+-----------+
| id | name    | age | info      |
+----+---------+-----+-----------+
|  2 | Laura   |  25 | NULL      |
|  5 | Evans   |  27 | secretary |
|  7 | Edison  |  28 | singer    |
| 11 | Beckham |  31 | police    |
+----+---------+-----+-----------+
```

结果显示 person 表中还有 4 条记录，执行 DELETE 语句删除这 4 条记录：

```
mysql> DELETE FROM person;
Query OK, 4 rows affected (0.00 sec)
```

语句执行完毕，查看执行结果：

```
mysql> SELECT * FROM person;
Empty set (0.00 sec)
```

查询结果为空，删除表中所有记录成功，现在 person 表中已经没有任何数据记录。



提 示

如果想删除表中的所有记录，还可以使用 TRUNCATE TABLE 语句，TRUNCATE 将直接删除原来的表并重新创建一个表，其语法结构为 TRUNCATE TABLE table_name。TRUNCATE 直接删除表而不是删除记录，因此执行速度比 DELETE 快。

17.4 查询数据

MySQL 从数据表中查询数据的基本语句为 SELECT 语句。SELECT 语句的基本格式是：

```
SELECT
        {* | <字段列表>}
```

```
        [
            FROM <表 1>,<表 2>...
            [WHERE <表达式>]
            [GROUP BY <group by definition>]
            [HAVING <expression> [{<operator> <expression>}...]]
            [ORDER BY <order by definition>]
            [LIMIT [<offset>,] <row count>]
        ]
SELECT [字段 1,字段 2,…,字段 n]
FROM [表或视图]
WHERE [查询条件];
```

- {* | <字段列表>}：包含星号通配符和字段列表。“*”表示查询所有的字段；“字段列表”表示查询指定的字段，字段列表至少包含一个子段名称，如果要查询多个字段，多个字段之间用逗号隔开，最后一个字段后不要加逗号。
- FROM <表 1>,<表 2>...：表 1 和表 2 表示查询数据的来源，可以是单个或者多个。
- WHERE 子句：可选项，如果选择该项，[查询条件]将限定查询行必须满足的查询条件。
- GROUP BY <字段>：该子句告诉 MySQL 如何显示查询出来的数据，并按照指定的字段分组。
- [ORDER BY <字段 >]：该子句告诉 MySQL 按什么样的顺序显示查询出来的数据。可以进行的排序有：升序(ASC)、降序（DESC）。
- [LIMIT [<offset>,] <row count>]：该子句告诉 MySQL 每次显示查询出来的数据条数。
- SELECT 的可选参数比较多。读者可能无法一下子完全理解，不要紧，接下来从最简单的开始，一步一步深入学习之后，就会对各个参数的作用有清晰的认识。

下面创建数据表 fruits，该表中包含了本章中需要用到的数据。

首先定义数据表。

```
CREATE TABLE fruits
(
    f_id    char(10)      NOT NULL,
    s_id    INT           NOT NULL,
    f_name  char(255)     NOT NULL,
    f_price decimal(8,2)  NOT NULL,
    PRIMARY KEY(f_id)
);
```

为了演示如何使用 SELECT 语句，需要插入数据。请读者插入如下数据：

```
mysql> INSERT INTO fruits (f_id, s_id, f_name, f_price)
    -> VALUES('a1', 101,'apple',5.2),
    -> ('b1',101,'blackberry', 10.2),
    -> ('bs1',102,'orange', 11.2),
    -> ('bs2',105,'melon',8.2),
```

```
    -> ('t1',102,'banana', 10.3),
    -> ('t2',102,'grape', 5.3),
    -> ('o2',103,'coconut', 9.2),
    -> ('c0',101,'cherry', 3.2),
    -> ('a2',103, 'apricot',2.2),
    -> ('l2',104,'lemon', 6.4),
    -> ('b2',104,'berry', 7.6),
    -> ('m1',106,'mango', 15.6),
    -> ('m2',105,'xbabay', 2.6),
    -> ('t4',107,'xbababa', 3.6),
    -> ('m3',105,'xxtt', 11.6),
    -> ('b5',107,'xxxx', 3.6);
```

使用 SELECT 语句查询 f_id 和 f_name 字段的数据。

```
mysql> SELECT f_id, f_name FROM fruits;
+------+------------+
| f_id | f_name     |
+------+------------+
| a1   | apple      |
| a2   | apricot    |
| b1   | blackberry |
| b2   | berry      |
| b5   | xxxx       |
| bs1  | orange     |
| bs2  | melon      |
| c0   | cherry     |
| l2   | lemon      |
| m1   | mango      |
| m2   | xbabay     |
| m3   | xxtt       |
| o2   | coconut    |
| t1   | banana     |
| t2   | grape      |
| t4   | xbababa    |
+------+------------+
16 rows in set (0.00 sec)
```

该语句的执行过程是，SELECT 语句决定了要查询的列值，在这里查询 f_id 和 f_name 两个字段的值，FROM 子句指定了数据的来源，这里指定数据表 fruits，因此返回结果为 fruits 表中 f_id 和 f_name 这两个字段下所有的数据。其显示顺序为添加到表中的顺序。

17.4.1 查询所有字段

1. 在 SELECT 语句中使用星号“*”通配符查询所有字段

SELECT 查询记录最简单的形式是从一个表中检索所有记录，实现的方法是使用星号（*）通配符指定查找所有的列的名称。语法格式如下：

```
SELECT * FROM 表名;
```

【例 17.13】从 fruits 表中检索所有字段的数据，SQL 语句如下：

```
mysql> SELECT * FROM fruits;
+------+------+------------+---------+
| f_id | s_id | f_name     | f_price |
+------+------+------------+---------+
| a1   | 101  | apple      |    5.20 |
| a2   | 103  | apricot    |    2.20 |
| b1   | 101  | blackberry |   10.20 |
| b2   | 104  | berry      |    7.60 |
| b5   | 107  | xxxx       |    3.60 |
| bs1  | 102  | orange     |   11.20 |
| bs2  | 105  | melon      |    8.20 |
| c0   | 101  | cherry     |    3.20 |
| l2   | 104  | lemon      |    6.40 |
| m1   | 106  | mango      |   15.60 |
| m2   | 105  | xbabay     |    2.60 |
| m3   | 105  | xxtt       |   11.60 |
| o2   | 103  | coconut    |    9.20 |
| t1   | 102  | banana     |   10.30 |
| t2   | 102  | grape      |    5.30 |
| t4   | 107  | xbababa    |    3.60 |
+------+------+------------+---------+
```

可以看到，使用星号通配符时，将返回所有列，列按照定义表的时候的顺序显示。

2. 在 SELECT 语句中指定所有字段

另外一种查询所有字段值的方法，根据前面 SELECT 语句格式，SELECT 关键字后面字段名为将要查找的数据。如果忘记了字段名称，可以使用 DESC 命令查看表的结构。有时候，可能表中的字段比较多，不一定能记得所有字段的名称，因此该方法有时候很不方便，不建议使用。例如查询 fruits 表中的所有数据，SQL 语句也可以书写如下：

```
SELECT f_id, s_id ,f_name, f_price FROM fruits;
```

查询结果与例 17.14 相同。



提　示

一般情况下，除非需要使用表中所有的字段数据，最好不要使用通配符“*”，使用通配符虽然可以节省输入查询语句的时间，但是获取不需要的列数据会降低查询的效率和所使用的应用程序的效率。通配符的优势是，当不知道所需要的列的名称时，可以获取它们。

17.4.2　查询指定字段

1．查询单个字段

查询表中的某一个字段，语法格式为：

```
SELECT 列名 FROM 表名;
```

【例 17.14】查询当前表中 f_name 列所有水果名称，SQL 语句如下：

```
SELECT f_name FROM fruits;
```

该语句使用 SELECT 声明从 fruits 表中获取名称为 f_name 字段下的所有水果名称，指定字段的名称紧跟在 SELECT 关键字之后，查询结果如下：

```
mysql> SELECT f_name FROM fruits;
+------------+
| f_name     |
+------------+
| apple      |
| apricot    |
| blackberry |
| berry      |
| xxxx       |
| orange     |
| melon      |
| cherry     |
| lemon      |
| mango      |
| xbabay     |
| xxtt       |
| coconut    |
| banana     |
| grape      |
| xbababa    |
+------------+
```

输出结果显示了 fruits 表中 f_name 字段下的所有数据。

2．查询多个字段

使用 SELECT 语句，可以获取多个字段下的数据，只需要在关键字 SELECT 后面指定要查找的字段的名称，不同字段名称之间用逗号（，）分隔开，最后一个字段后面不需要加逗号，语法格

式如下：

```
SELECT 字段名1,字段名2,…,字段名n  FROM 表名;
```

【例 17.15】从 fruits 表中获取 f_name 和 f_price 两列，SQL 语句如下：

```
SELECT f_name, f_price FROM fruits;
```

该语句使用 SELECT 声明从 fruits 表中获取名称为 f_name 和 f_price 两个字段下的所有水果名称和价格，两个字段之间用逗号分隔开，查询结果如下：

```
mysql> SELECT f_name, f_price FROM fruits;
+------------+---------+
| f_name     | f_price |
+------------+---------+
| apple      |   5.20  |
| apricot    |   2.20  |
| blackberry |  10.20  |
| berry      |   7.60  |
| xxxx       |   3.60  |
| orange     |  11.20  |
| melon      |   8.20  |
| cherry     |   3.20  |
| lemon      |   6.40  |
| mango      |  15.60  |
| xbabay     |   2.60  |
| xxtt       |  11.60  |
| coconut    |   9.20  |
| banana     |  10.30  |
| grape      |   5.30  |
| xbababa    |   3.60  |
+------------+---------+
```

输出结果显示了 fruits 表中 f_name 和 f_price 两个字段下的所有数据。



提 示

MySQL 中的 SQL 语句是不区分大小写的，因此 SELECT 和 select 作用是相同的，但是，许多开发人员习惯将关键字使用大写，而数据列和表名使用小写，读者也应该养成一个良好的编程习惯，这样写出来的代码便于阅读和维护。

17.4.3 查询指定记录

数据库中包含大量的数据，根据特殊要求，可能只需查询表中的指定数据，即对数据进行过滤。在 SELECT 语句中通过 WHERE 子句，对数据进行过滤，语法格式为：

```
SELECT 字段名1,字段名2,…,字段名n
```

```
FROM 表名
WHERE 查询条件
```

在 WHERE 子句中，MySQL 提供了一系列的条件判断符，如表 17-1 所示。

表 17-1　WHERE 条件判断符

操作符	说明
=	相等
<>，!=	不相等
<	小于
<=	小于或者等于
>	大于
>=	大于或者等于
BETWEEN	位于两值之间

【例 17.16】查询价格为 10.2 元的水果的名称，SQL 语句如下：

```
SELECT f_name, f_price
FROM fruits
WHERE f_price = 10.2;
```

该语句使用 SELECT 声明从 fruits 表中获取价格等于 10.2 的水果的数据，从查询结果可以看到价格是 10.2 元的水果的名称 blackberry，其他的均不满足查询条件，查询结果如下：

```
mysql> SELECT f_name, f_price
    -> FROM fruits
    -> WHERE f_price = 10.2;
+------------+---------+
| f_name     | f_price |
+------------+---------+
| blackberry |   10.20  |
+------------+---------+
```

本例采用了简单的相等过滤，查询一个指定列 f_price 具有值 10.20。

相等还可以用来比较字符串。

【例 17.17】查找名称为 apple 的水果的价格，SQL 语句如下：

```
SELECT f_name, f_price
FROM fruits
WHERE f_name = 'apple';
```

该语句使用 SELECT 声明从 fruits 表中获取名称为 apple 的水果的价格，从查询结果可以看到只有名称为 apple 行被返回，其他的均不满足查询条件。

```
mysql> SELECT f_name, f_price
```

```
    -> FROM fruits
    -> WHERE f_name = 'apple';
+--------+---------+
| f_name | f_price |
+--------+---------+
| apple  |   5.20  |
+--------+---------+
```

【例 17.18】查询价格小于 10.00 元的水果的名称，SQL 语句如下：

```
SELECT f_name, f_price
FROM fruits
WHERE f_price < 10.00;
```

该语句使用 SELECT 声明从 fruits 表中获取价格低于 10.00 元的水果名称，即 f_price 小于 10.00 的水果信息被返回，查询结果如下：

```
mysql> SELECT f_name, f_price
    -> FROM fruits
    -> WHERE f_price < 10.00;
+---------+---------+
| f_name  | f_price |
+---------+---------+
| apple   |   5.20  |
| apricot |   2.20  |
| berry   |   7.60  |
| xxxx    |   3.60  |
| melon   |   8.20  |
| cherry  |   3.20  |
| lemon   |   6.40  |
| xbabay  |   2.60  |
| coconut |   9.20  |
| grape   |   5.30  |
| xbababa |   3.60  |
+---------+---------+
```

可以看到查询结果中所有记录的 f_price 字段的值均小于 10.00 元。而大于 10.00 元的记录没有被返回。

17.4.4 带 IN 关键字的查询

IN 操作符用来查询满足指定条件范围内的记录。使用 IN 操作符时，将所有检索条件用括号括起来，检索条件用逗号分隔开，只要满足条件范围内的一个值即为匹配项。

【例 17.19】查询 s_id 为 101 和 102 的记录，SQL 语句如下：

```
SELECT s_id,f_name, f_price
FROM fruits
WHERE s_id IN (101,102)
ORDER BY f_name;
```

查询结果如下：

```
+------+-----------+---------+
| s_id | f_name     | f_price |
+------+-----------+---------+
|  101 | apple      |    5.20  |
|  102 | banana     |   10.30  |
|  101 | blackberry |   10.20  |
|  101 | cherry     |    3.20  |
|  102 | grape      |    5.30  |
|  102 | orange     |   11.20  |
+------+-----------+---------+
```

相反地，可以是用关键字 NOT 来检索不在条件范围内的记录。

【例 17.20】查询所有 s_id 不等于 101 也不等于 102 的记录，SQL 语句如下：

```
SELECT s_id,f_name, f_price
FROM fruits
WHERE s_id NOT IN (101,102)
ORDER BY f_name;
```

查询结果如下：

```
+------+---------+---------+
| s_id | f_name  | f_price  |
+------+---------+---------+
|  103 | apricot |    2.20  |
|  104 | berry   |    7.60  |
|  103 | coconut |    9.20 |
|  104 | lemon   |    6.40  |
|  106 | mango   |   15.60  |
|  105 | melon   |    8.20  |
|  107 | xbababa |    3.60  |
|  105 | xbabay  |    2.60   |
|  105 | xxtt    |   11.60  |
|  107 | xxxx    |    3.60  |
+------+---------+---------+
```

可以看到，该语句在 IN 关键字前面加上了 NOT 关键字，这使得查询的结果与前面一个的结果正好相反，前面检索了 s_id 等于 101 和 102 的记录，而这里所要求的查询的记录中的 s_id 字段

值不等于这两个值中的任一个。

17.4.5 带 BETWEEN AND 的范围查询

BETWEEN AND 用来查询某个范围内的值，该操作符需要两个参数，即范围的开始值和结束值，如果记录的字段值满足指定的范围查询条件，则这些记录被返回。

【例 17.21】查询价格在 2.00 元到 10.5 元之间水果名称和价格，SQL 语句如下：

```
SELECT f_name, f_price FROM fruits WHERE f_price BETWEEN 2.00 AND 10.20;
```

查询结果如下：

```
mysql> SELECT f_name, f_price
    -> FROM fruits
    -> WHERE f_price BETWEEN 2.00 AND 10.20;
+------------+---------+
| f_name     | f_price |
+------------+---------+
| apple      |   5.20  |
| apricot    |   2.20  |
| blackberry |  10.20  |
| berry      |   7.60  |
| xxxx       |   3.60  |
| melon      |   8.20  |
| cherry     |   3.20  |
| lemon      |   6.40  |
| xbabay     |   2.60  |
| coconut    |   9.20  |
| grape      |   5.30  |
| xbababa    |   3.60  |
+------------+---------+
```

可以看到，返回结果包含了价格从 2.00 元到 10.20 元之间的字段值，并且端点值 10.20 也包括在返回结果中，即 BETWEEN 匹配范围中所有值，包括开始值和结束值。

BETWEEN AND 操作符前可以加关键字 NOT，表示指定范围之外的值，如果字段值不满足指定的范围内的值，则这些记录被返回。

【例 17.22】查询价格在 2.00 元到 10.5 元之外的水果名称和价格，SQL 语句如下：

```
SELECT f_name, f_price
FROM fruits
WHERE f_price NOT BETWEEN 2.00 AND 10.20;
```

查询结果如下：

```
+--------+---------+
```

```
| f_name | f_price |
+--------+---------+
| orange |   11.20 |
| mango  |   15.60 |
| xxtt   |   11.60 |
| banana |   10.30 |
+--------+---------+
```

由结果可以看到，返回的记录只有 f_price 字段大于 10.20 的， f_price 字段小于 2.00 的记录也满足查询条件，因此如果表中有 f_price 字段小于 2.00 的记录，也应当作为查询结果。

17.4.6 带 LIKE 的字符匹配查询

在前面的检索操作中，讲述了如何查询多个字段的记录，如何进行比较查询或者查询一个条件范围内的记录，如果要查找所有的包含字符“ge”的水果名称，该如何查找呢？简单的比较操作在这里已经行不通了，在这里，需要使用通配符进行匹配查找，通过创建查找模式对表中的数据进行比较。执行这个任务的关键字是 LIKE。

通配符是一种在 SQL 的 WHERE 条件子句中拥有特殊意思的字符，SQL 语句中支持多种通配符，可以和 LIKE 一起使用的通配符有‘%’和‘_’。

1. 百分号通配符‘%’，匹配任意长度的字符，甚至包括零字符

【例 17.23】查找所有以‘b’字母开头的水果，SQL 语句如下：

```
SELECT f_id, f_name
FROM fruits
WHERE f_name LIKE 'b%';
```

查询结果如下：

```
+------+------------+
| f_id | f_name     |
+------+------------+
| b1   | blackberry |
| b2   | berry      |
| t1   | banana     |
+------+------------+
```

该语句查询的结果返回所有以‘b’开头的水果的 id 和 name， ‘%’告诉 MySQL，返回所有 f_name 字段以字母‘b’开头的记录，不管‘b’后面有多少个字符。

在搜索匹配时通配符‘%’可以放在不同位置。

【例 17.24】在 fruits 表中，查询 f_name 中包含字母‘g’的记录，SQL 语句如下：

```
SELECT f_id, f_name
FROM fruits
WHERE f_name LIKE '%g%';
```

查询结果如下：

```
+------+--------+
| f_id | f_name |
+------+--------+
| bs1  | orange |
| m1   | mango  |
| t2   | grape  |
+------+--------+
```

该语句查询包含字符串中包含字母‘g’的水果名称，只要名字中有字符‘g’，而前面或后面不管有多少个字符，都满足查询的条件。

【例 17.25】查询以‘b’开头，并以‘y’结尾的水果的名称，SQL 语句如下：

```
SELECT f_name
FROM fruits
WHERE f_name LIKE 'b%y';
```

查询结果如下：

```
+------------+
| f_name     |
+------------+
| blackberry |
| berry      |
+------------+
```

通过以上查询结果，可以看到，‘%’用于匹配在指定的位置的任意数目的字符。

2. 下划线通配符‘_’，一次只能匹配任意一个字符

另一个非常有用的通配符是下划线通配符‘_’，该通配符的用法和‘%’相同，区别是‘%’匹配多个字符，而‘_’只匹配任意单个字符，如果要匹配多个字符，则需要使用相同个数的‘_’。

【例 17.26】在 fruits 表中，查询以字母‘y’结尾，且‘y’前面只有 4 个字母的记录，SQL 语句如下：

```
SELECT f_id, f_name FROM fruits WHERE f_name LIKE '_ _ _ _y';
```

查询结果如下：

```
+------+--------+
| f_id | f_name |
+------+--------+
| b2   | berry  |
+------+--------+
```

从结果可以看到，以‘y’结尾且前面只有 4 个字母的记录只有一条。其他记录的 f_name 字

段也有以‘y’结尾的，但其总的字符串长度不为 5，因此不在返回结果中。

17.4.7 查询空值

创建数据表的时候，设计者可以指定某列中是否可以包含空值（NULL），空值不同于 0，也不同于空字符串，空值一般表示数据未知、不适用或将在以后添加数据。在 SELECT 语句中使用 IS NULL 子句，可以查询某字段内容为空记录。

【例 17.27】查询 customers 表中 c_email 为空的记录的 c_id、c_name 和 c_email 字段值，SQL 语句如下：

```
SELECT c_id, c_name,c_email FROM customers WHERE c_email IS NULL;
```

查询结果如下：

```
mysql> SELECT c_id, c_name,c_email FROM customers WHERE c_email IS NULL;
+-------+-----------+---------+
| c_id  | c_name    | c_email |
+-------+-----------+---------+
| 10003 | Netbhood  | NULL    |
+-------+-----------+---------+
```

可以看到，显示 customers 表中字段 c_email 的值为 NULL 的记录，满足查询条件。

与 IS NULL 相反的是 NOT IS NULL，该关键字查找字段不为空的记录。

【例 17.28】查询 customers 表中 c_email 不为空的记录的 c_id、c_name 和 c_email 字段值，SQL 语句如下：

```
SELECT c_id, c_name,c_email FROM customers WHERE c_email IS NOT NULL;
```

查询结果如下：

```
mysql> SELECT c_id, c_name,c_email FROM customers WHERE c_email IS NOT NULL;
+-------+---------+-------------------+
| c_id  | c_name  | c_email           |
+-------+---------+-------------------+
| 10001 | RedHook | LMing@163.com     |
| 10002 | Stars   | Jerry@hotmail.com |
| 10004 | JOTO    | sam@hotmail.com   |
+-------+---------+-------------------+
```

可以看到，查询出来的记录的 c_email 字段都不为空值。

17.4.8 带 AND 的多条件查询

使用 SELECT 查询时，可以增加查询的限制条件，这样可以使查询的结果更加精确。MySQL 在 WHERE 子句中使用 AND 操作符限定只有满足所有查询条件的记录才会被返回。可以使用 AND 连接两个甚至多个查询条件，多个条件表达式之间用 AND 分开。

【例 17.29】在 fruits 表中查询 s_id = 101，并且 f_price 大于 5 的记录价格和名称，SQL 语句如下：

```
SELECT s_id, f_price, f_name FROM fruits WHERE s_id = '101' AND f_price >=5;
```

查询结果如下：

```
mysql> SELECT s_id, f_price, f_name
    -> FROM fruits
    -> WHERE s_id = '101' AND f_price >= 5;
+------+---------+------------+
| s_id | f_price | f_name     |
+------+---------+------------+
| 101  |    5.20 | apple      |
| 101  |   10.20 | blackberry |
+------+---------+------------+
```

前面的语句检索了 s_id=101 的水果供应商所有价格大于等于 5 元的水果名称和价格。WHERE 子句中的条件分为两部分，AND 关键字指示 MySQL 返回所有同时满足两个条件的行。即使是 s_id=101 的水果供应商提供的水果，价格<5 或者 s_id 不等于'101'的水果供应商的水果不管其价格为多少，均不是要查询的结果。


提 示

上述例子的 WHERE 子句中只包含了一个 AND 语句，把两个过滤条件组合在一起。实际上可以添加多个 AND 过滤条件，增加条件的同时增加一个 AND 关键字。

【例 17.30】在 fruits 表中查询 s_id = 101 或者 102，并且 f_price 大于 5，f_name='apple'的记录的价格和名称，SQL 语句如下：

```
SELECT f_id, f_price, f_name FROM fruits
WHERE s_id IN('101', '102') AND f_price >= 5 AND f_name = 'apple';
```

查询结果如下：

```
mysql> SELECT f_id, f_price, f_name FROM fruits
     -> WHERE s_id IN('101','102') AND f_price >= 5 AND f_name = 'apple';
+------+---------+--------+
| s_id | f_price | f_name |
+------+---------+--------+
| 101  |    5.20 | apple  |
+------+---------+--------+
```

可以看到返回符合查询条件的记录只有一条。

17.4.9　带 OR 的多条件查询

与 AND 相反，在 WHERE 声明中使用 OR 操作符，表示只需要满足其中一个条件的记录即可

返回。OR 也可以连接两个甚至多个查询条件，多个条件表达式之间用 OR 分开。

【例 17.31】查询 s_id=101 或者 s_id=102 的水果供应商的 f_price 和 f_name，SQL 语句如下：

```
SELECT s_id,f_name, f_price FROM fruits WHERE s_id = 101 OR s_id = 102;
```

查询结果如下：

```
mysql> SELECT s_id,f_name, f_price
    -> FROM fruits
    -> WHERE s_id = 101 OR s_id = 102;
+------+------------+---------+
| s_id | f_name     | f_price |
+------+------------+---------+
|  101 | apple      |    5.20 |
|  101 | blackberry |   10.20 |
|  102 | orange     |   11.20 |
|  101 | cherry     |    3.20 |
|  102 | banana     |   10.30 |
|  102 | grape      |    5.30 |
+------+------------+---------+
```

结果显示查询了 s_id=101 和 s_id=102 的供应商提供的水果名称和价格，OR 操作符告诉 MySQL，检索的时候只需要满足其中的一个条件，不需要全部都满足，如果这里使用 AND 的话，将检索不到符合条件的数据。

在这里，也可以使用 IN 操作符实现与 OR 相同的功能，下面的例子可进行说明。

【例 17.32】查询 s_id=101 或者 s_id=102 的水果供应商的 f_name 和 f_price，SQL 语句如下：

```
SELECT s_id,f_name, f_price FROM fruits WHERE s_id IN(101,102);
```

查询结果如下：

```
mysql> SELECT s_id,f_name, f_price
    -> FROM fruits
    -> WHERE s_id IN(101,102);
+------+------------+---------+
| s_id | f_name     | f_price |
+------+------------+---------+
|  101 | apple      |    5.20 |
|  101 | blackberry |   10.20 |
|  102 | orange     |   11.20 |
|  101 | cherry     |    3.20 |
|  102 | banana     |   10.30 |
|  102 | grape      |    5.30 |
+------+------------+---------+
```

在这里可以看到，OR 操作符和 IN 操作符使用后的结果是一样的，它们可以实现相同的功能。

但是使用 IN 操作符使得检索语句更加简洁明了，并且 IN 执行的速度要快于 OR，更重要的是，使用 IN 操作符后，可以执行更加复杂的嵌套查询（后面章节将会讲述）。



提 示

OR 可以和 AND 一起使用，但是在使用时要注意两者的优先级，由于 AND 的优先级高于 OR，因此先对 AND 两边的操作数进行操作，再与 OR 中的操作数结合。

17.4.10 查询结果不重复

从前面的例子可以看到，SELECT 查询返回所有匹配的行。例如查询 fruits 表中所有的 s_id，其结果为：

```
+------+
| s_id |
+------+
|  101 |
|  103 |
|  101 |
|  104 |
|  107 |
|  102 |
|  105 |
|  101 |
|  104 |
|  106 |
|  105 |
|  105 |
|  103 |
|  102 |
|  102 |
|  107 |
+------+
```

可以看到查询结果返回了 16 条记录，其中有一些重复的 s_id 值，有时，出于对数据分析的要求，需要消除重复的记录值，如何使查询结果没有重复呢？在 SELECT 语句中可以使用 DISTINCT 关键字指示 MySQL 消除重复的记录值。语法格式为：

```
SELECT DISTINCT 字段名 FROM 表名;
```

【例 17.33】查询 fruits 表中 s_id 字段的值，并返回 s_id 字段值不得重复，SQL 语句如下：

```
SELECT DISTINCT s_id FROM fruits;
```

查询结果如下：

```
mysql> SELECT DISTINCT s_id FROM fruits;
```

```
+-------+
| s_id |
+-------+
|  101 |
|  103 |
|  104 |
|  107 |
|  102 |
|  105 |
|  106 |
+-------+
```

可以看到这次查询结果只返回了 7 条记录的 s_id 值，而不再有重复的值，SELECT DISTINCE s_id 告诉 MySQL 只返回不同的 s_id 值。

17.4.11 对查询结果排序

从前面的查询结果，读者会发现有些字段的值是没有任何顺序的，MySQL 中可以通过 SELECT 使用 ORDER BY 子句对查询的结果进行排序。

1. 单列排序

例如，查询 f_name 字段，查询结果如下：

```
mysql> SELECT f_name FROM fruits;
+------------+
| f_name     |
+------------+
| apple      |
| apricot    |
| blackberry |
| berry      |
| xxxx       |
| orange     |
| melon      |
| cherry     |
| lemon      |
| mango      |
| xbabay     |
| xxtt       |
| coconut    |
| banana     |
| grape      |
| xbababa    |
+------------+
```

可以看到，查询的数据并没有以一种特定的顺序显示，如果没有对它们进行排序，将根据它们插入到数据表中的顺序来显示。

下面使用 ORDER BY 子句对指定的列数据进行排序。

【例 17.34】查询 fruits 表的 f_name 字段值，并对其进行排序，SQL 语句如下：

```
mysql> SELECT f_name FROM fruits ORDER BY f_name;
+------------+
| f_name     |
+------------+
| apple      |
| apricot    |
| banana     |
| berry      |
| blackberry |
| cherry     |
| coconut    |
| grape      |
| lemon      |
| mango      |
| melon      |
| orange     |
| xbababa    |
| xbabay     |
| xxtt       |
| xxxx       |
+------------+
```

该语句查询的结果和前面的语句相同，不同的是，通过指定 ORDER BY 子句，MySQL 对查询的 f_name 列的数据按字母表的顺序进行了升序排序。

2. 多列排序

有时，需要根据多列值进行排序。例如，如果要显示一个学生列表，可能会有多个学生的姓氏是相同的，因此还需要根据学生的名进行排序。对多列数据进行排序，只要将需要排序的列之间用逗号隔开。

【例 17.35】查询 fruits 表中的 f_name 和 f_price 字段，先按 f_name 排序，再按 f_price 排序，SQL 语句如下：

```
SELECT f_name, f_price FROM fruits ORDER BY f_name, f_price;
```

查询结果如下：

```
mysql> SELECT f_name, f_price FROM fruits ORDER BY f_name, f_price;
+------------+---------+
```

```
| f_name      | f_price |
+------------+---------+
| apple       |   5.20  |
| apricot     |   2.20  |
| banana      |  10.30  |
| berry       |   7.60  |
| blackberry|  10.20  |
| cherry      |   3.20  |
| coconut     |   9.20  |
| grape       |   5.30  |
| lemon       |   6.40  |
| mango       |  15.60  |
| melon       |   8.20  |
| orange      |  11.20  |
| xbababa     |   3.60  |
| xbabay      |   2.60  |
| xxtt        |  11.60  |
| xxxx        |   3.60  |
+------------+---------+
```



在对多列进行排序的时候，首先排序的第一列必须有相同的列值，才会对第二列进行排序，如果第一列数据中所有值都是唯一的，将不再对第二列进行排序。

3. 指定排序方向

默认情况下，查询数据按字母升序进行排序（从 A 到 Z），但数据的排序并不仅限于此，还可以使用 ORDER BY 对查询结果进行降序排序（从 Z 到 A），这通过关键字 DESC 实现，下面的例子表明了如何降序排列。

【例 17.36】查询 fruits 表中的 f_name 和 f_price 字段，对结果按 f_price 降序方式排序，SQL 语句如下：

```
SELECT f_name, f_price FROM fruits ORDER BY f_price DESC;
```

查询结果如下：

```
mysql> SELECT f_name,  f_price FROM fruits ORDER BY f_price DESC;
+------------+---------+
| f_name      | f_price |
+------------+---------+
| mango       |  15.60 |
| xxtt        |  11.60 |
| orange      |  11.20 |
| banana      |  10.30 |
```

```
| blackberry |   10.20 |
| coconut    |    9.20 |
| melon      |    8.20 |
| berry      |    7.60 |
| lemon      |    6.40 |
| grape      |    5.30 |
| apple      |    5.20 |
| xxxx       |    3.60 |
| xbababa    |    3.60 |
| cherry     |    3.20 |
| xbabay     |    2.60 |
| apricot    |    2.20 |
+------------+---------+
```



提 示

与 DESC 相反的是 ASC（升序排序），将字段列中的数据按字母表顺序升序排序，实际上在排序的时候 ASC 是作为默认的排序方式，所以加不加都可以。

也可以对多列进行不同的顺序排序。

【例 17.37】查询 fruits 表，先按 f_price 降序排序，再按 f_name 字段升序排序，SQL 语句如下：

```
SELECT f_price, f_name FROM fruits ORDER BY f_price DESC, f_name;
```

查询结果如下：

```
mysql> SELECT f_price, f_name FROM fruits ORDER BY f_price DESC, f_name;
+---------+------------+
| f_price | f_name     |
+---------+------------+
|   15.60 | mango      |
|   11.60 | xxtt       |
|   11.20 | orange     |
|   10.30 | banana     |
|   10.20 | blackberry |
|    9.20 | coconut    |
|    8.20 | melon      |
|    7.60 | berry      |
|    6.40 | lemon      |
|    5.30 | grape      |
|    5.20 | apple      |
|    3.60 | xbababa    |
|    3.60 | xxxx       |
|    3.20 | cherry     |
|    2.60 | xbabay     |
|    2.20 | apricot    |
+---------+------------+
```

DESC 排序方式只应用到直接位于其前面的字段上，由结果可以看到。


提 示

DESC 关键字只对其前面的列降序排列，在这里只对 f_price 排序，而并没有对 f_name 进行排序，因此，f_price 按降序排序，而 f_name 列仍按升序排序。如果要对多列都进行降序排序，必须要在每一列的列名后面加 DESC 关键字。

17.5 实战演练——数据表综合应用案例

此案例根据不同的条件对表进行查询操作，涉及的表如表 17-2~表 17-5 所示。

表 17-2 employee 表结构

字段名	字段说明	数据类型	主键	外键	非空	唯一	自增
e_no	员工编号	INT	是	否	是	是	否
e_name	员工姓名	VARCHAR(100)	否	否	是	否	否
e_gender	员工性别	CHAR(2)	否	否	是	否	否
dept_no	部门编号	INT	否	是	是	否	否
e_job	职位	VARCHAR(100)	否	否	是	否	否
e_salary	薪水	SMALLINT	否	否	是	否	否
hireDate	入职日期	DATE	否	否	是	否	否

表 17-3 dept 表结构

字段名	字段说明	数据类型	主键	外键	非空	唯一	自增
d_no	部门编号	INT	是	是	是	是	是
d_name	部门名称	VARCHAR(50)	否	否	是	否	否
d_location	部门地址	VARCHAR(100)	否	否	否	否	否

表 17-4 employee 表中的记录

e_no	e_name	e_gender	dept_no	e_job	e_salary	hireDate
1001	SMITH	m	20	CLERK	800	2005-11-12
1002	ALLEN	f	30	SALESMAN	1600	2003-05-12
1003	WARD	f	30	SALESMAN	1250	2003-05-12
1004	JONES	m	20	MANAGER	2975	1998-05-18
1005	MARTIN	m	30	SALESMAN	1250	2001-06-12
1006	BLAKE	f	30	MANAGER	2850	1997-02-15
1007	CLARK	m	10	MANAGER	2450	2002-09-12
1008	SCOTT	m	20	ANALYST	3000	2003-05-12
1009	KING	f	10	PRESIDENT	5000	1995-01-01
1010	TURNER	f	30	SALESMAN	1500	1997-10-12
1011	ADAMS	m	20	CLERK	1100	1999-10-05
1012	JAMES	f	30	CLERK	950	2008-06-15

表 17-5　dept 表中的记录

d_no	d_name	d_location
10	ACCOUNTING	ShangHai
20	RESEARCH	BeiJing
30	SALES	ShenZhen
40	OPERATIONS	FuJian

案例操作过程如下。

01 创建数据表 employee 和 dept。

```
CREATE TABLE dept
(
    d_no        INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    d_name      VARCHAR(50)NOT NULL,
    d_location  VARCHAR(100)
);
```

由于 employee 表中的 dept_no 依赖于父表 dept 的主键 d_no，因此需要先创建 dept 表，然后创建 employee 表。

```
CREATE TABLE employee
(
    e_no        INT NOT NULL PRIMARY KEY,
    e_name      VARCHAR(100) NOT NULL,
    e_gender    CHAR(2) NOT NULL,
    dept_no    INT NOT NULL,
    e_job       VARCHAR(100) NOT NULL,
    e_salary   SMALLINT NOT NULL,
    hireDate   DATE,
    CONSTRAINT dno_fk FOREIGN KEY(dept_no)
    REFERENCES dept(d_no)
);
```

02 将指定记录分别插入两个表中。

向 dept 表中插入数据，SQL 语句如下：

```
INSERT INTO dept
VALUES (10, 'ACCOUNTING', 'ShangHai'),
       (20, 'RESEARCH ', 'BeiJing '),
       (30, 'SALES ', 'ShenZhen '),
       (40, 'OPERATIONS ', 'FuJian ');
```

向 employee 表中插入数据，SQL 语句如下：

```
INSERT INTO employee
VALUES (1001, 'SMITH', 'm',20, 'CLERK',800,'2005-11-12'),
       (1002, 'ALLEN', 'f',30, 'SALESMAN', 1600,'2003-05-12'),
       (1003, 'WARD', 'f',30, 'SALESMAN', 1250,'2003-05-12'),
       (1004, 'JONES', 'm',20, 'MANAGER', 2975,'1998-05-18'),
       (1005, 'MARTIN', 'm',30, 'SALESMAN', 1250,'2001-06-12'),
       (1006, 'BLAKE', 'f',30, 'MANAGER', 2850,'1997-02-15'),
       (1007, 'CLARK', 'm',10, 'MANAGER', 2450,'2002-09-12'),
       (1008, 'SCOTT', 'm',20, 'ANALYST', 3000,'2003-05-12'),
       (1009, 'KING', 'f',10, 'PRESIDENT', 5000,'1995-01-01'),
       (1010, 'TURNER', 'f',30, 'SALESMAN', 1500,'1997-10-12'),
       (1011, 'ADAMS', 'm',20, 'CLERK', 1100,'1999-10-05'),
       (1012, 'JAMES', 'm',30, 'CLERK', 950,'2008-06-15');
```

03 在 employee 表中，查询所有记录的 e_no、e_name 和 e_salary 字段值。

```
SELECT e_no, e_name, e_salary FROM employee;
```

执行结果如下：

```
mysql> SELECT e_no, e_name, e_salary FROM employee;
+------+--------+----------+
| e_no | e_name | e_salary |
+------+--------+----------+
| 1001 | SMITH  |   800    |
| 1002 | ALLEN  |   1600   |
| 1003 | WARD   |   1250   |
| 1004 | JONES  |   2975   |
| 1005 | MARTIN |   1250   |
| 1006 | BLAKE  |   2850   |
| 1007 | CLARK  |   2450   |
| 1008 | SCOTT  |   3000   |
| 1009 | KING   |   5000   |
| 1010 | TURNER |   1500   |
| 1011 | ADAMS  |   1100   |
| 1012 | JAMES  |   950    |
+------+--------+----------+
12 rows in set (0.00 sec)
```

04 在 employee 表中，查询 dept_no 等于 10 和 20 的所有记录。

```
SELECT * FROM employee WHERE dept_no IN (10, 20);
```

执行结果如下：

```
mysql> SELECT * FROM employee WHERE dept_no IN (10, 20);
+------+--------+---------+---------+----------+----------+-----------+
| e_no | e_name | e_gender | dept_no | e_job    | e_salary | hireDate  |
+------+--------+---------+---------+----------+----------+-----------+
| 1001 | SMITH  | m        |     20  | CLERK     |    800    | 2005-11-12 |
| 1004 | JONES  | m        |     20  | MANAGER  |    2975   | 1998-05-18 |
| 1007 | CLARK  | m        |     10  | MANAGER  |   2450   | 2002-09-12 |
| 1008 | SCOTT  | m        |     20  | ANALYST  |    3000   | 2003-05-12 |
| 1009 | KING   | f         |     10   | PRESIDENT |     5000   | 1995-01-01  |
| 1011 | ADAMS  | m        |     20  | CLERK     |   1100   | 1999-10-05 |
+------+--------+---------+---------+----------+----------+-----------+
6 rows in set (0.00 sec)
```

05 在 employee 表中，查询工资范围在 800 到 2500 之间的员工信息。

```
SELECT * FROM employee WHERE e_salary BETWEEN 800 AND 2500;
```

执行结果如下：

```
mysql>  SELECT * FROM employee WHERE e_salary BETWEEN 800 AND 2500;
+------+--------+----------+---------+----------+----------+-----------+
| e_no | e_name | e_gender | dept_no | e_job     | e_salary | hireDate |
+------+--------+----------+---------+----------+----------+-----------+
| 1001 | SMITH  | m         |      20  | CLERK     |    800     | 2005-11-12 |
| 1002 | ALLEN  | f         |      30  | SALESMAN |   1600     | 2003-05-12 |
| 1003 | WARD   | f         |      30  | SALESMAN |   1250     | 2003-05-12 |
| 1005 | MARTIN | m          |      30  | SALESMAN |    1250    | 2001-06-12 |
| 1007 | CLARK  | m         |      10  | MANAGER  |    2450    | 2002-09-12 |
| 1010 | TURNER | f          |      30  | SALESMAN|    1500    | 1997-10-12 |
| 1011 | ADAMS  | m         |      20  | CLERK     |   1100    | 1999-10-05 |
| 1012 | JAMES  | m         |      30  | CLERK     |   950     | 2008-06-15 |
+------+--------+----------+---------+----------+----------+-----------+
8 rows in set (0.00 sec)
```

06 在 employee 表中，查询部门编号为 20 的部门中的员工信息。

```
SELECT * FROM employee WHERE dept_no = 20;
```

执行结果如下：

```
mysql> SELECT * FROM employee WHERE dept_no = 20;
+------+--------+----------+---------+----------+----------+------------+
| e_no | e_name | e_gender | dept_no | e_job    | e_salary | hireDate    |
+------+--------+----------+---------+----------+----------+------------+
| 1001 | SMITH  | m         |      20  | CLERK    |     800    | 2005-11-12 |
| 1004 | JONES  | m         |      20  | MANAGER |    2975    | 1998-05-18 |
```

```
| 1008 | SCOTT  | m         |    20  | ANALYST |    3000   | 2003-05-12 |
| 1011 | ADAMS  | m         |    20  | CLERK   |    1100   | 1999-10-05 |
+------+--------+----------+---------+---------+----------+------------+
4 rows in set (0.00 sec)
```

07 在 employee 表中，查询每个部门最高工资的员工信息。

```
SELECT dept_no, MAX(e_salary) FROM employee GROUP BY dept_no;
```

执行结果如下：

```
mysql> SELECT dept_no, MAX(e_salary) FROM employee GROUP BY dept_no;
+---------+---------------+
| dept_no | MAX(e_salary) |
+---------+---------------+
|     10  |          5000 |
|     20  |          3000 |
|     30  |          2850 |
+---------+---------------+
3 rows in set (0.00 sec)
```

08 查询员工 BLAKE 所在部门和部门所在地。

```
SELECT d_no, d_location  FROM dept WHERE d_no=
        (SELECT dept_no FROM employee WHERE e_name='BLAKE');
```

执行结果如下：

```
mysql> SELECT d_no, d_location
    -> FROM dept WHERE d_no=
    ->                (SELECT dept_no FROM employee WHERE e_name='BLAKE');
+------+------------+
| d_no | d_location |
+------+------------+
|   30 | ShenZhen   |
+------+------------+
1 row in set (0.00 sec)
```

09 使用连接查询，查询所有员工的部门和部门信息。

```
SELECT e_no, e_name, dept_no, d_name,d_location
FROM employee, dept WHERE dept.d_no=employee.dept_no;
```

执行结果如下：

```
mysql> SELECT e_no, e_name, dept_no, d_name,d_location
    -> FROM employee, dept WHERE dept.d_no=employee.dept_no;
+------+--------+---------+------------+------------+
```

```
| e_no | e_name | dept_no | d_name      | d_location |
+------+--------+---------+------------+------------+
| 1001 | SMITH  |      20 | RESEARCH   | BeiJing    |
| 1002 | ALLEN  |      30 | SALES      | ShenZhen   |
| 1003 | WARD   |      30 | SALES      | ShenZhen   |
| 1004 | JONES  |      20 | RESEARCH   | BeiJing    |
| 1005 | MARTIN |      30 | SALES      | ShenZhen   |
| 1006 | BLAKE  |      30 | SALES      | ShenZhen   |
| 1007 | CLARK  |      10 | ACCOUNTING | ShangHai   |
| 1008 | SCOTT  |      20 | RESEARCH   | BeiJing    |
| 1009 | KING   |      10 | ACCOUNTING | ShangHai   |
| 1010 | TURNER |      30 | SALES      | ShenZhen   |
| 1011 | ADAMS  |      20 | RESEARCH   | BeiJing    |
| 1012 | JAMES  |      30 | SALES      | ShenZhen   |
+------+--------+---------+------------+------------+
12 rows in set (0.00 sec)
```

10 在 employee 表中，计算每个部门各有多少名员工。

```
SELECT dept_no, COUNT(*) FROM employee GROUP BY dept_no;
```

执行结果如下：

```
mysql> SELECT dept_no, COUNT(*) FROM employee GROUP BY dept_no;
+---------+----------+
| dept_no | COUNT(*) |
+---------+----------+
|      10 |        2 |
|      20 |        4 |
|      30 |        6 |
+---------+----------+
3 rows in set (0.00 sec)
```

11 在 employee 表中，计算不同类型职工的总工资数。

```
SELECT e_job, SUM(e_salary) FROM employee GROUP BY e_job;
```

执行结果如下：

```
mysql> SELECT e_job, SUM(e_salary) FROM employee GROUP BY e_job;
+-----------+---------------+
| e_job     | SUM(e_salary) |
+-----------+---------------+
| ANALYST   |          3000 |
| CLERK     |          2850 |
| MANAGER   |          8275 |
```

```
| PRESIDENT |          5000 |
| SALESMAN  |          5600 |
+-----------+---------------+
5 rows in set (0.00 sec)
```

12 在 employee 表中，计算不同部门的平均工资。

```
SELECT dept_no, AVG(e_salary) FROM employee GROUP BY dept_no;
```

执行结果如下：

```
mysql> SELECT dept_no, AVG(e_salary) FROM employee GROUP BY dept_no;
+---------+---------------+
| dept_no | AVG(e_salary) |
+---------+---------------+
|      10 |     3725.0000 |
|      20 |     1968.7500 |
|      30 |     1566.6667 |
+---------+---------------+
3 rows in set (0.00 sec)
```

13 在 employee 表中，查询工资低于 1500 的员工信息。

```
SELECT * FROM employee WHERE e_salary < 1500;
```

执行过程如下：

```
mysql> SELECT * FROM employee WHERE e_salary < 1500;
+------+--------+----------+---------+----------+---------+------------+
| e_no | e_name | e_gender | dept_no | e_job    | e_salary| hireDate   |
+------+--------+----------+---------+----------+---------+------------+
| 1001 | SMITH  | m        |      20 | CLERK    |     800 | 2005-11-12 |
| 1003 | WARD   | f        |      30 | SALESMAN |    1250 | 2003-05-12 |
| 1005 | MARTIN | m        |      30 | SALESMAN |    1250 | 2001-06-12 |
| 1011 | ADAMS  | m        |      20 | CLERK    |    1100 | 1999-10-05 |
| 1012 | JAMES  | m        |      30 | CLERK    |     950 | 2008-06-15 |
+------+--------+----------+---------+----------+---------+------------+
5 rows in set (0.00 sec)
```

14 在 employee 表中，将查询记录先按部门编号由高到低排列，再按员工工资由高到低排列。

```
SELECT e_name,dept_no, e_salary
FROM employee ORDER BY dept_no DESC, e_salary DESC;
```

执行过程如下：

```
mysql> SELECT e_name,dept_no, e_salary
    -> FROM employee ORDER BY dept_no DESC, e_salary DESC;
```

```
+--------+---------+----------+
| e_name | dept_no | e_salary |
+--------+---------+----------+
| BLAKE  |      30 |     2850 |
| ALLEN  |      30 |     1600 |
| TURNER |      30 |     1500 |
| WARD   |      30 |     1250 |
| MARTIN |      30 |     1250 |
| JAMES  |      30 |      950 |
| SCOTT  |      20 |     3000 |
| JONES  |      20 |     2975 |
| ADAMS  |      20 |     1100 |
| SMITH  |      20 |      800 |
| KING   |      10 |     5000 |
| CLARK  |      10 |     2450 |
+--------+---------+----------+
12 rows in set (0.00 sec)
```

15 在 employee 表中，查询员工姓名以字母 A 或 S 开头的员工的信息。

```
SELECT * FROM employee WHERE e_name REGEXP '^[as]';
```

执行过程如下：

```
mysql> SELECT * FROM employee WHERE e_name REGEXP '^[as]';
+------+--------+----------+---------+----------+----------+------------+
| e_no | e_name | e_gender | dept_no | e_job    | e_salary | hireDate   |
+------+--------+----------+---------+----------+----------+------------+
| 1001 | SMITH  | m        |      20 | CLERK    |      800 | 2005-11-12 |
| 1002 | ALLEN  | f        |      30 | SALESMAN |     1600 | 2003-05-12 |
| 1008 | SCOTT  | m        |      20 | ANALYST  |     3000 | 2003-05-12 |
| 1011 | ADAMS  | m        |      20 | CLERK    |     1100 | 1999-10-05 |
+------+--------+----------+---------+----------+----------+------------+
4 rows in set (0.00 sec)
```

16 在 employee 表中，查询到目前为止，工龄大于等于 15 年的员工信息。

```
SELECT * FROM employee where YEAR(CURDATE()) -YEAR(hireDate) >= 15;
```

执行过程如下：

```
mysql> SELECT * FROM employee where YEAR(CURDATE()) -YEAR(hireDate) >= 15;
+------+--------+----------+---------+----------+----------+------------+
| e_no | e_name | e_gender | dept_no | e_job    | e_salary | hireDate   |
+------+--------+----------+---------+----------+----------+------------+
| 1004 | JONES  | m        |      20 | MANAGER  |     2975 | 1998-05-18 |
```

```
| 1005 | MARTIN | m        |      30 | SALESMAN  |    1250   | 2001-06-12 |
| 1006 | BLAKE  | f        |      30 | MANAGER   |    2850   | 1997-02-15 |
| 1009 | KING   | f        |      10 | PRESIDENT |    5000   | 1995-01-01 |
| 1010 | TURNER | f        |      30 | SALESMAN  |    1500   | 1997-10-12 |
| 1011 | ADAMS  | m        |      20 | CLERK     |    1100   | 1999-10-05 |
+------+--------+----------+---------+-----------+-----------+------------+
6 rows in set (0.01 sec)
```

17.6 高手私房菜

技巧 1：插入记录时可以不指定字段名称吗？

不管使用哪种 INSERT 语法，都必须给出 VALUES 的正确数目。如果提供字段名，则不必给每个字段提供一个值；如果不提供字段名，则必须为每个字段提供一个值，否则将产生一条错误消息。如果要在 INSERT 操作中省略某些字段，这些字段需要满足一定条件：该列定义为允许空值；或者表定义时给出默认值，如果不给出值，将使用默认值。

技巧 2：更新或者删除表时必须指定 WHERE 子句吗？

在前面章节中可以看到，所有的 UPDATE 和 DELETE 语句全都在 WHERE 子句指定了条件。如果省略 WHERE 子句，则 UPDATE 或 DELETE 将被应用到表中所有的行。因此，除非确实打算更新或者删除所有记录，否则绝对要注意使用不带 WHERE 子句的 UPDATE 或 DELETE 语句。建议在对表进行更新和删除操作之前，使用 SELECT 语句确认需要删除的记录，以免造成无法挽回的结果。

技巧 3：什么时候使用引号？

在查询的时候，会看到在 WHERE 子句中使用条件，有的值加上了单引号，而有的值未加。单引号用来限定字符串，如果将值与字符串类型列进行比较，则需要限定引号，用来与数值进行比较的值不需要用引号。

技巧 4：为什么使用通配符格式正确，却没有查找出符合条件的记录？

MySQL 中存储字符串数据时，可能会不小心把两端带有空格的字符串保存到记录中，而在查看表中记录时，MySQL 不能明确地显示空格，数据库操作者不能直观地确定字符串两端是否有空格，例如，使用 LIKE '%e' 匹配以字母 e 结尾的水果的名称，如果字母 e 后面多了一个空格。则 LIKE 语句不能将该记录查找出来。解决的方法是使用 TRIM 函数，将字符串两端的空格删除之后再进行匹配。

17.7 经典习题

创建数据表 pet，并对表进行插入、更新与删除操作，pet 表结构如表 17-6 所示。

（1）首先创建数据表 pet，使用不同的方法将表 17-7 中的记录插入到 pet 表中。
（2）使用 UPDATE 语句将名称为 Fang 的狗的主人改为 Kevin。
（3）将没有主人的宠物的 owner 字段值都改为 Duck。

（4）删除已经死亡的宠物记录。

（5）删除所有表中的记录。

表 17-6　pet 表结构

字段名	字段说明	数据类型	主键	外键	非空	唯一	自增
name	宠物名称	VARCHAR(20)	否	否	是	否	否
owner	宠物主人	VARCHAR(20)	否	否	否	否	否
species	种类	VARCHAR(20)	否	否	是	否	否
sex	性别	CHAR(1)	否	否	是	否	否
birth	出生日期	YEAR	否	否	是	否	否
death	死亡日期	YEAR	否	否	否	否	否

表 17-7　pet 表中记录

name	owner	Species	sex	birth	death
Fluffy	Harold	Cat	f	2003	2010
Claws	Gwen	cat	m	2004	NULL
Buffy	NULL	dog	f	2009	NULL
Fang	Benny	dog	m	2000	NULL
Bowser	Diane	dog	m	2003	2009
Chirpy	NULL	bird	f	2008	NULL

第 18 章　数据库的备份与还原

尽管采取了一些管理措施来保证数据库的安全，但是不确定的意外情况总是有可能造成数据的损失，例如意外的停电、管理员不小心的操作失误都可能会造成数据的丢失。保证数据安全的最重要的一个措施是确保对数据进行定期备份。如果数据库中的数据丢失或者出现错误，可以使用备份的数据进行恢复，这样就尽可能地降低了意外原因导致的损失。MySQL 提供了多种方法对数据进行备份和恢复。本章将介绍数据备份、数据恢复、数据迁移和数据导入导出的相关知识。

本章学习目标

- 了解什么是数据备份
- 掌握各种数据备份的方法
- 掌握各种数据恢复的方法
- 掌握数据库迁移的方法
- 掌握表的导入和导出方法
- 熟练掌握综合案例中数据备份与恢复的方法和技巧

18.1　数据备份

数据备份是数据库管理员非常重要的工作之一。系统意外崩溃或者硬件的损坏都可能导致数据库的丢失，因此 MySQL 管理员应该定期地备份数据库，使得在意外情况发生时，尽可能减少损失。本节将介绍数据备份的 3 种方法。

18.1.1　使用 MySQLdump 命令备份

MySQLdump 是 MySQL 提供的一个非常有用的数据库备份工具。MySQLdump 命令执行时，可以将数据库备份成一个文本文件，该文件中实际上包含了多个 CREATE 和 INSERT 语句，使用这些语句可以重新创建表和插入数据。

MySQLdump 备份数据库语句的基本语法格式如下：

```
mysqldump  -u user -h host -p password dbname[tbname, [tbname...]]> filename.sql
```

user 表示用户名称；host 表示登录用户的主机名称；password 为登录密码；dbname 为需要备份的数据库名称；tbname 为 dbname 数据库中需要备份的数据表，可以指定多个需要备份的表；右箭头符号“>”告诉 MySQLdump 将备份数据表的定义和数据写入备份文件；filename.sql 为备份文件的名称。

1. 使用 MySQLdump 备份单个数据库中的所有表

【例 18.1】使用 MySQLdump 命令备份数据库中的所有表，执行过程如下：

为了更好理解 MySQLdump 工具如何工作，本章给出一个完整的数据库例子。首先登录 MySQL，按下面数据库结构创建 booksDB 数据库和各个表，并插入数据记录。数据库和表定义如下：

```
CREATE DATABASE booksDB;
use booksDB;

CREATE TABLE books
(
bk_id INT NOT NULL PRIMARY KEY,
bk_title VARCHAR(50) NOT NULL,
copyright YEAR NOT NULL
);
INSERT INTO books
VALUES (11078, 'Learning MySQL', 2010),
(11033, 'Study Html', 2011),
(11035, 'How to use php', 2003),
(11072, 'Teach yourself javascript', 2005),
(11028, 'Learning C++', 2005),
(11069, 'MySQL professional', 2009),
(11026, 'Guide to MySQL 5.7', 2008),
(11041, 'Inside VC++', 2011);

CREATE TABLE authors
(
auth_id INT NOT NULL PRIMARY KEY,
auth_name VARCHAR(20),
auth_gender CHAR(1)
);
INSERT INTO authors
VALUES (1001, 'WriterX' ,'f'),
(1002, 'WriterA' ,'f'),
(1003, 'WriterB' ,'m'),
(1004, 'WriterC' ,'f'),
(1011, 'WriterD' ,'f'),
(1012, 'WriterE' ,'m'),
(1013, 'WriterF' ,'m'),
(1014, 'WriterG' ,'f'),
(1015, 'WriterH' ,'f');

CREATE TABLE authorbook
(
auth_id INT NOT NULL,
bk_id INT NOT NULL,
PRIMARY KEY (auth_id, bk_id),
```

```
FOREIGN KEY (auth_id) REFERENCES authors (auth_id),
FOREIGN KEY (bk_id) REFERENCES books (bk_id)
);

INSERT INTO authorbook
VALUES (1001, 11033), (1002, 11035), (1003, 11072), (1004, 11028),
(1011, 11078), (1012, 11026), (1012, 11041), (1014, 11069);
```

完成数据插入后打开操作系统命令行输入窗口，输入备份命令如下：

```
C:\ >mysqldump -u root -p booksdb > C:/backup/booksdb_20160301.sql
Enter password: **
```

输入密码之后，MySQL 便对数据库进行了备份，在 C:\backup 文件夹下面查看刚才备份过的文件，使用文本查看器打开文件可以看到其部分文件内容大致如下：

```
-- MySQL dump 10.13  Distrib 5.7.10, for Win32 (x86)
--
-- Host: localhost    Database: booksDB
-- ------------------------------------------------------
-- Server version   5.7.10

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_
CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE=
'NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `authorbook`
--

DROP TABLE IF EXISTS `authorbook`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `authorbook` (
  `auth_id` int(11) NOT NULL,
  `bk_id` int(11) NOT NULL,
  PRIMARY KEY (`auth_id`,`bk_id`),
  KEY `bk_id` (`bk_id`),
```

```
  CONSTRAINT `authorbook_ibfk_1` FOREIGN KEY (`auth_id`)
  REFERENCES `authors` (`auth_id`),
  CONSTRAINT `authorbook_ibfk_2` FOREIGN KEY (`bk_id`)
REFERENCES `books` (`bk_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `authorbook`
--

LOCK TABLES `authorbook` WRITE;
/*!40000 ALTER TABLE `authorbook` DISABLE KEYS */;
INSERT INTO `authorbook` VALUES (1012,11026),(1004,11028),(1001,11033),
(1002,11035),(1012, 11041),(1014,11069),(1003,11072),(1011,11078);
/*!40000 ALTER TABLE `authorbook` ENABLE KEYS */;
UNLOCK TABLES;
…
…省略部分内容
…
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
-- Dump completed on 2011-08-18 10:44:08
```

可以看到，备份文件包含了一些信息，文件开头首先表明了备份文件使用的 MySQLdump 工具的版本号；然后是备份账户的名称和主机信息，以及备份的数据库的名称，最后是 MySQL 服务器的版本号，在这里为 5.7.10。

备份文件接下来的部分是一些 SET 语句，这些语句将一些系统变量值赋给用户定义变量，以确保被恢复的数据库的系统变量和原来备份时的变量相同，例如：

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
```

该 SET 语句将当前系统变量 character_set_client 的值赋给用户定义变量@old_character_set_client。其他变量与此类似。

备份文件的最后几行 MySQL 使用 SET 语句恢复服务器系统变量原来的值，例如：

```
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

该语句将用户定义的变量@old_character_set_client 中保存的值赋给实际的系统变量

character_set_client。

备份文件中的“--”字符开头的行为注释语句；以“/*!”开头、“*/”结尾的语句为可执行的 MySQL 注释，这些语句可以被 MySQL 执行，但在其他数据库管理系统将被作为注释忽略，这可以提高数据库的可移植性。

另外注意到，备份文件开始的一些语句以数字开头，这些数字代表了 MySQL 版本号，该数字告诉我们，这些语句只有在指定的 MySQL 版本或者比该版本高的情况下才能执行。例如 40101，表明这些语句只有在 MySQL 版本号为 4.01.01 或者更高的条件下才可以被执行。

2. 使用 MySQLdump 备份数据库中的某个表

在前面 MySQLdump 语法中介绍过，MySQLdump 还可以备份数据中的某个表，其语法格式为：

```
mysqldump  -u user -h host -p dbname  [tbname, [tbname...]] > filename.sql
```

tbname 表示数据库中的表名，多个表名之间用空格隔开。

备份表和备份数据库中所有表的语句中不同的地方在于，要在数据库名称 dbname 之后指定需要备份的表名称。

【例 18.2】备份 booksDB 数据库中的 books 表，输入语句如下：

```
mysqldump -u root -p booksDB books > C:/backup/books_20160301.sql
```

该语句创建名称为 books_20160301.sql 的备份文件，文件中包含了前面介绍的 SET 语句等内容，不同的是，该文件只包含 books 表的 CREATE 和 INSERT 语句。

3. 使用 MySQLdump 备份多个数据库

如果要使用 MySQLdump 备份多个数据库，需要使用--databases 参数。备份多个数据库的语句格式如下：

```
mysqldump  -u user -h host -p --databases  [dbname, [dbname...]] > filename.sql
```

使用--databases 参数之后，必须指定至少一个数据库的名称，多个数据库名称之间用空格隔开。

【例 18.3】使用 MySQLdump 备份 booksDB 和 test 数据库，输入语句如下：

```
mysqldump -u root -p --databases  booksDB test> C:\backup\books_testDB_
20160301.sql
```

该语句创建名称为 books_testDB_20160301.sql 的备份文件，文件中包含了创建两个数据库 booksDB 和 test_db 所必须的所有语句。

另外，使用--all-databases 参数可以备份系统中所有的数据库，语句如下：

```
mysqldump  -u user -h host -p --all-databases > filename.sql
```

使用参数--all-databases 时，不需要指定数据库名称。

【例 18.4】使用 MySQLdump 备份服务器中的所有数据库，输入语句如下：

```
mysqldump  -u root -p --all-databases > C:/backup/alldbinMySQL.sql
```

该语句创建名称为 alldbinMySQL.sql 的备份文件，文件中包含了对系统中所有数据库的备份信息。


提 示

如果在服务器上进行备份，并且表均为 MyISAM 表，应考虑使用 MySQLhotcopy，因为可以更快地进行备份和恢复。

MySQLdump 还有一些其他选项可以用来制定备份过程，例如--opt 选项，该选项将打开--quick 、--add-locks、--extended-insert 等多个选项。使用--opt 选项可以提供最快速的数据库转储。

MySQLdump 其他常用选项如下：

- --add-drop-database：在每个 CREATE DATABASE 语句前添加 DROP DATABASE 语句。
- --add-drop-tables：在每个 CREATE TABLE 语句前添加 DROP TABLE 语句。
- --add-locking：用 LOCK TABLES 和 UNLOCK TABLES 语句引用每个表转储。重载转储文件时插入得更快。
- --all--database，-A：转储所有数据库中的所有表。与使用--database 选项相同，在命令行中命名所有数据库。
- --comments[=0|1]：如果设置为 0，禁止转储文件中的其他信息，例如程序版本、服务器版本和主机。--skip-comments 与--comments=0 的结果相同。默认值为 1，即包括额外信息。
- --compact：产生少量输出。该选项禁用注释并启用--skip-add-drop-tables、--no-set-names、--skip-disable-keys 和--skip-add-locking 选项。
- --compatible=name：产生与其他数据库系统或旧的 MySQL 服务器更兼容的输出。值可以为 ansi、MySQL323、MySQL40、postgresql、oracle、mssql、db2、maxdb、no_key_options、no_tables_options 或者 no_field_options。
- --complete-insert，-c：使用包括列名的完整的 INSERT 语句。
- ---debug[=debug_options]，-# [debug_options]：写调试日志。
- --delete，-D：导入文本文件前清空表。
- --default-character-set=charset：使用 charsetas 默认字符集。如果没有指定，MySQLdump 使用 utf8。
- --delete-master-logs：在主复制服务器上，完成转储操作后删除二进制日志。该选项自动启用-master-data。
- --extended-insert，-e：使用包括几个 VALUES 列表的多行 INSERT 语法。这样使转储文件更小，重载文件时可以加速插入。
- --flush-logs，-F：开始转储前刷新 MySQL 服务器日志文件。该选项要求 RELOAD 权限。
- --force，-f：在表转储过程中，即使出现 SQL 错误也继续。
- --lock-all-tables，-x：对所有数据库中的所有表加锁。在整体转储过程中通过全局锁定来实现。该选项自动关闭--single-transaction 和--lock-tables。
- --lock-tables，-l：开始转储前锁定所有表。用 READ LOCAL 锁定表以允许并行插入 MyISAM 表。对于事务表（例如 InnoDB 和 BDB），--single-transaction 是一个更好的选项，因为它根本不需要锁定表。
- --no-create-db，-n：该选项禁用 CREATE DATABASE /*!32312 IF NOT EXISTS*/ db_name

语句，如果给出--database 或--all--database 选项，则包含到输出中。

- --no-create-info，-t：只导出数据，而不添加 CREATE TABLE 语句。
- --no-data，-d：不写表的任何行信息，只转储表的结构。
- --opt：该选项是速记，等同于指定--add-drop-tables--add-locking、--create-option、--disable-keys--extended-insert、--lock-tables-quick 和--set-charset。它可以快速进行转储操作并产生一个能很快装入 MySQL 服务器的转储文件。该选项默认开启，但可以用--skip-opt 禁用。要想禁用使用-opt 启用的选项，可以使用--skip 形式，例如--skip-add-drop-tables 或--skip-quick。
- --password[=password]，-p[password]：当连接服务器时使用的密码。如果使用短选项形式（-p），选项和密码之间不能有空格。如果在命令行中--password 或-p 选项后面没有密码值，则提示输入一个密码。
- --port=port_num，-P port_num：用于连接的 TCP/IP 端口号。
- --protocol={TCP | SOCKET | PIPE | MEMORY}：使用的连接协议。
- --replace，-r --replace 和—ignore：控制替换或复制唯一键值已有记录的输入记录的处理。如果指定--replace，新行替换有相同的唯一键值的已有行；如果指定--ignore，复制已有的唯一键值的输入行被跳过。如果不指定这两个选项，当发现一个复制键值时会出现一个错误，并且忽视文本文件的剩余部分。
- --silent，-s：沉默模式。只有出现错误时才输出。
- --socket=path，-S path：当连接 localhost 时使用的套接字文件（为默认主机）。
- --user=user_name，-u user_name：当连接服务器时 MySQL 使用的用户名。
- --verbose，-v：冗长模式。打印出程序操作的详细信息。
- --version，-V：显示版本信息并退出。
- --xml，-X：产生 XML 输出。

MySQLdump 提供许多选项，包括用于调试和压缩的，在这里只是列举最有用的。运行帮助命令 MySQLdump --help，可以获得特定版本的完整选项列表。



提 示

如果运行 MySQLdump 没有--quick 或--opt 选项，MySQLdump 在转储结果前将整个结果集装入内存。如果转储大数据库，可能会出现问题。该选项默认启用，但可以用--skip-opt 禁用。如果使用最新版本的 MySQLdump 程序备份数据，并用于恢复到比较旧版本的 MySQL 服务器中，则不要使用--opt 或-e 选项。

18.1.2 直接复制整个数据库目录

因为 MySQL 表保存为文件方式，所以可以直接复制 MySQL 数据库的存储目录及文件进行备份。MySQL 的数据库目录位置不一定相同，在 Windows 平台下，MySQL 5.7 存放数据库的目录通常默认为 C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.7\data 或者其他用户自定义目录；在 Linux 平台下，数据库目录位置通常为/var/lib/MySQL/，不同 Linux 版本下目录会有不同，读者应在自己使用的平台下查找该目录。

这是一种简单、快速、有效的备份方式。要想保持备份的一致性，备份前需要对相关表执行 LOCK TABLES 操作，然后对表执行 FLUSH TABLES。这样当复制数据库目录中的文件时，允许

其他客户继续查询表。需要 FLUSH TABLES 语句来确保开始备份前将所有激活的索引页写入硬盘。当然，也可以停止 MySQL 服务再进行备份操作。

这种方法虽然简单，但并不是最好的方法。因为这种方法对 InnoDB 存储引擎的表不适用。使用这种方法备份的数据最好恢复到相同版本的服务器中，不同的版本可能不兼容。


提 示

在 MySQL 版本号中，第一个数字表示主版本号，主版本号相同的 MySQL 数据库文件格式相同。

18.1.3 使用 MySQLhotcopy 工具快速备份

MySQLhotcopy 是一个 Perl 脚本，最初由 Tim Bunce 编写并提供。它使用 LOCK TABLES、FLUSH TABLES 和 cp 或 scp 来快速备份数据库。它是备份数据库或单个表的最快的途径，但它只能运行在数据库目录所在的机器上，并且只能备份 MyISAM 类型的表。MySQLhotcopy 在 Unix 系统中运行。

MySQLhotcopy 命令语法格式如下：

```
mysqlhotcopy db_name_1, ... db_name_n  /path/to/new_directory
```

db_name_1,…,db_name_n 分别为需要备份的数据库的名称；/path/to/new_directory 指定备份文件目录。

【例 18.5】使用 MySQLhotcopy 备份 test 数据库到/usr/backup 目录下，输入语句如下：

```
mysqlhotcopy  -u root -p test /usr/backup
```

要想执行 MySQLhotcopy，必须可以访问备份的表文件，具有那些表的 SELECT 权限、RELOAD 权限（以便能够执行 FLUSH TABLES）和 LOCK TABLES 权限。


提 示

MySQLhotcopy 只是将表所在的目录复制到另一个位置，只能用于备份 MyISAM 和 ARCHIVE 表。备份 InnoDB 类型的数据表时会出现错误信息。由于它复制本地格式的文件，故也不能移植到其他硬件或操作系统下。

18.2 数据恢复

管理人员操作的失误、计算机故障以及其他意外情况，都会导致数据的丢失和破坏。当数据丢失或意外破坏时，可以通过恢复已经备份的数据尽量减少数据丢失和破坏造成的损失。本节将介绍数据恢复的方法。

18.2.1 使用 MySQL 命令恢复

对于已经备份的包含 CREATE、INSERT 语句的文本文件，可以使用 MySQL 命令导入到数据库中。本小节将介绍 MySQL 命令导入 sql 文件的方法。

备份的 sql 文件中包含 CREATE、INSERT 语句（有时也会有 DROP 语句）。MySQL 命令可

以直接执行文件中的这些语句。其语法如下：

```
mysql -u user -p [dbname] < filename.sql
```

user 是执行 backup.sql 中语句的用户名；-p 表示输入用户密码；dbname 是数据库名。如果 filename.sql 文件为 MySQLdump 工具创建的包含创建数据库语句的文件，执行的时候不需要指定数据库名。

【例 18.6】使用 MySQL 命令将 C:\backup\booksdb_20160301.sql 文件中的备份导入到数据库中，输入语句如下：

```
mysql -u root -p booksDB < C:/backup/booksdb_20160301.sql
```

执行该语句前，必须先在 MySQL 服务器中创建 booksDB 数据库，如果不存在恢复过程将会出错。命令执行成功之后 booksdb_20160301.sql 文件中的语句就会在指定的数据库中恢复以前的表。

如果已经登录 MySQL 服务器，还可以使用 source 命令导入 sql 文件。source 语句语法如下：

```
source filename
```

【例 18.7】使用 root 用户登录到服务器，然后使用 source 导入本地的备份文件 booksdb_20160301.sql，输入语句如下：

```
--选择要恢复到的数据库
mysql> use booksDB;
Database changed

--使用 source 命令导入备份文件
mysql> source C:\backup\booksDB_20160301.sql
```

命令执行后，会列出备份文件 booksDB_20160301.sql 中每一条语句的执行结果。source 命令执行成功后，booksDB_20160301.sql 中的语句会全部导入到现有数据库中。


提 示

执行 source 命令前，必须使用 use 语句选择数据库。不然，恢复过程中会出现“ERROR 1046 (3D000): No database selected”的错误。

18.2.2 直接复制到数据库目录

如果数据库通过复制数据库文件备份，可以直接复制备份的文件到 MySQL 数据目录下实现恢复。通过这种方式恢复时，必须保存备份数据的数据库和待恢复的数据库服务器的主版本号相同。而且这种方式只对 MyISAM 引擎的表有效，对于 InnoDB 引擎的表不可用。

执行恢复以前关闭 MySQL 服务，将备份的文件或目录覆盖 MySQL 的 data 目录，启动 MySQL 服务。对于 Linux/Unix 操作系统来说，复制完文件需要将文件的用户和组更改为 MySQL 运行的用户和组，通常用户是 MySQL，组也是 MySQL。

18.2.3 MySQLhotcopy 快速恢复

MySQLhotcopy 备份后的文件也可以用来恢复数据库，在 MySQL 服务器停止运行时，将备份的数据库文件复制到 MySQL 存放数据的位置（MySQL 的 data 文件夹），重新启动 MySQL 服务即可。如果以根用户执行该操作，必须指定数据库文件的所有者，输入语句如下：

```
chown -R mysql.mysql /var/lib/mysql/dbname
```

【例 18.8】从 MySQLhotcopy 复制的备份恢复数据库，输入语句如下：

```
cp -R  /usr/backup/test usr/local/mysql/data
```

执行完该语句，重启服务器，MySQL 将恢复到备份状态。



如果需要恢复的数据库已经存在，则在使用 DROP 语句删除已经存在的数据库之后，恢复才能成功。另外 MySQL 不同版本之间必须兼容，恢复之后的数据才可以使用。

18.3 数据库迁移

数据库迁移就是把数据从一个系统移动到另一个系统上。数据迁移有以下原因：

（1）需要安装新的数据库服务器。
（2）MySQL 版本更新。
（3）数据库管理系统的变更（如从 Microsoft SQL Server 迁移到 MySQL）。

本小节将讲解数据库迁移的方法。

18.3.1 相同版本的 MySQL 数据库之间的迁移

相同版本的 MySQL 数据库之间的迁移就是在主版本号相同的 MySQL 数据库之间进行数据库移动。迁移过程其实就是在源数据库备份和目标数据库恢复过程的组合。

在讲解数据库备份和恢复时，已经知道最简单的方式是通过复制数据库文件目录，但是此种方法只适用于 MyISAM 引擎的表。而对于 InnoDB 表，不能用直接复制文件的方式备份数据库，因此最常用和最安全的方式是使用 MySQLdump 命令导出数据，然后在目标数据库服务器使用 MySQL 命令导入。

【例 18.9】将 www.abc.com 主机上的 MySQL 数据库全部迁移到 www.bcd.com 主机上。在 www.abc.com 主机上执行的命令如下：

```
mysqldump -h www.bac.com -uroot -ppassword dbname |
mysql -h www.bcd.com -uroot -ppassword
```

MySQLdump 导出的数据直接通过管道符“|”，传给 MySQL 命令导入到主机 www.bcd.com 数据库中，dbname 为需要迁移的数据库名称，如果要迁移全部的数据库，可使用参数--all-databases。

18.3.2 不同版本的 MySQL 数据库之间的迁移

因为数据库升级等原因，需要将较旧版本 MySQL 数据库中的数据迁移到较新版本的数据库中。MySQL 服务器升级时，需要先停止服务，然后卸载旧版本，并安装新版的 MySQL，这种更新方法很简单，如果想保留旧版本中的用户访问控制信息，则需要备份 MySQL 中的 MySQL 数据库，在新版本 MySQL 安装完成之后，重新读入 MySQL 备份文件中的信息。

旧版本与新版本的 MySQL 可能使用不同的默认字符集，例如 MySQL 4.x 中大多使用 latin1 作为默认字符集，而 MySQL 5.x 的默认字符集为 utf8。如果数据库中有中文数据的，迁移过程中需要对默认字符集进行修改，不然可能无法正常显示结果。

新版本会对旧版本有一定兼容性。从旧版本的 MySQL 向新版本的 MySQL 迁移时，对于 MyISAM 引擎的表，可以直接复制数据库文件，也可以使用 MySQLhotcopy 工具、MySQLdump 工具。对于 InnoDB 引擎的表，一般只能使用 MySQLdump 将数据导出。然后使用 MySQL 命令导入到目标服务器上。从新版本向旧版本 MySQL 迁移数据时要特别小心，最好使用 MySQLdump 命令导出，然后导入目标数据库中。

18.3.3 不同数据库之间的迁移

不同类型的数据库之间的迁移，是指把 MySQL 的数据库转移到其他类型的数据库，例如从 MySQL 迁移到 ORACLE，从 ORACLE 迁移到 MySQL，从 MySQL 迁移到 sqlserver 等。

迁移之前，需要了解不同数据库的架构，比较它们之间的差异。不同数据库中定义相同类型的数据的关键字可能会不同。例如，MySQL 中日期字段分为 DATE 和 TIME 两种，而 ORACLE 日期字段只有 DATE。另外，数据库厂商并没有完全按照 SQL 标准来设计数据库系统，导致不同的数据库系统的 SQL 语句有差别。例如，MySQL 几乎完全支持标准 SQL 语言，而 Microsoft SQL Server 使用的是 T-SQL 语言，T-SQL 中有一些非标准的 SQL 语句，因此在迁移时必须对这些语句进行语句映射处理。

数据库迁移可以使用一些工具，例如在 Windows 系统下，可以使用 MyODBC 实现 MySQL 和 SQL Server 之间的迁移。MySQL 官方提供的工具 MySQL Migration Toolkit 也可以在不同数据库间进行数据迁移。

18.4 表的导出和导入

有时会需要将 MySQL 数据库中的数据导出到外部存储文件中，MySQL 数据库中的数据可以导出成 sql 文本文件、xml 文件或者 html 文件。同样这些导出文件也可以导入到 MySQL 数据库中。本小节将介绍数据导出和导入的常用方法。

18.4.1 使用 SELECTI…INTO OUTFILE 导出文本文件

MySQL 数据库导出数据时，允许使用包含导出定义的 SELECT 语句进行数据的导出操作。该文件被创建到服务器主机上，因此必须拥有文件写入权限（FILE 权限），才能使用此语法。“SELECT...INTO OUTFILE 'filename'”形式的 SELECT 语句可以把被选择的行写入一个文件中，filename 不能是一个已经存在的文件。SELECT...INTO OUTFILE 语句基本格式如下：

```
SELECT  columnlist   FROM  table  WHERE  condition   INTO  OUTFILE  'filename'
[OPTIONS]

--OPTIONS 选项
   FIELDS  TERMINATED BY 'value'
FIELDS  [OPTIONALLY] ENCLOSED BY 'value'
FIELDS  ESCAPED BY 'value'
LINES  STARTING BY 'value'
LINES  TERMINATED BY 'value'
```

可以看到 SELECT columnlist FROM table WHERE condition 为一个查询语句，查询结果返回满足指定条件的一条或多条记录；INTO OUTFILE 语句的作用就是把前面 SELECT 语句查询出来的结果导出到名称为“filename”的外部文件中。[OPTIONS]为可选参数选项，OPTIONS 部分的语法包括 FIELDS 和 LINES 子句，其可能的取值有：

- FIELDS TERMINATED BY ‘value’：设置字段之间的分隔字符，可以为单个或多个字符，默认情况下为制表符 ‘\t’。
- FIELDS [OPTIONALLY] ENCLOSED BY ‘value’：设置字段的包围字符，只能为单个字符，如果使用了 OPTIONALLY 则只有 CHAR 和 VERCHAR 等字符数据字段被包括。
- FIELDS ESCAPED BY ‘value’：设置如何写入或读取特殊字符，只能为单个字符，即设置转义字符，默认值为 ‘\’。
- LINES STARTING BY ‘value’：设置每行数据开头的字符，可以为单个或多个字符，默认情况下不使用任何字符。
- LINES TERMINATED BY ‘value’：设置每行数据结尾的字符，可以为单个或多个字符，默认值为 ‘\n’。

FIELDS 和 LINES 两个子句都是自选的，但是如果两个都被指定了，FIELDS 必须位于 LINES 的前面。

SELECT...INTO OUTFILE 语句可以非常快速地把一个表转储到服务器上。如果想要在服务器主机之外的部分客户主机上创建结果文件，不能使用 SELECT...INTO OUTFILE。在这种情况下，应该在客户主机上使用比如“MySQL –e "SELECT ..." > file_name”的命令来生成文件。

SELECT...INTO OUTFILE 是 LOAD DATA INFILE 的补语。用于语句的 OPTIONS 部分的语法包括部分 FIELDS 和 LINES 子句，这些子句与 LOAD DATA INFILE 语句同时使用。

【例 18.10】使用 SELECT...INTO OUTFILE 将 test 数据库中的 person 表中的记录导出到文本文件，输入命令如下：

```
SELECT *  FROM test.person INTO OUTFILE "C:\person0.txt";
```

由于指定了 INTO OUTFILE 子句，SELECT 将查询出来的 3 个字段的值保存到 C:\person0.txt 文件中，打开文件内容如下：

```
1    Green    21   Lawyer
2    Suse     22   dancer
3    Mary     24   Musician
```

```
4    Willam  20  sports man
5    Laura   25  \N
6    Evans   27  secretary
7    Dale    22  cook
8    Edison  28  singer
9    Harry   21  magician
10   Harriet 19  pianist
```

可以看到默认情况下，MySQL 使用制表符“\t”分隔不同的字段，字段没有被其他字符括起来。另外，在 Windows 平台下，使用记事本打开该文件，显示的格式与这里并不相同，这是因为 Windows 系统下回车换行符为“\r\n”，默认换行符为“\n”，因此会在 person.txt 中可能看到类似黑色方块的字符，所有的记录也会在同一行显示。

另外，注意到第 5 行中有一个字段值为“\N”，这表示该字段的值为 NULL。默认情况下，如果遇到 NULL 值，将会返回“\N”代表空值，反斜线“\”表示转义字符，如果使用 ESCAPED BY 选项，则 N 前面为指定的转义字符。

【例 18.11】使用 SELECT...INTO OUTFILE 将 test 数据库中的 person 表中的记录导出到文本文件，使用 FIELDS 选项和 LINES 选项，要求字段之间使用逗号“,”间隔，所有字段值用双引号括起来，定义转义字符为单引号“\'”，执行的命令如下：

```
SELECT * FROM  test.person INTO OUTFILE "C:\person1.txt"
FIELDS
TERMINATED BY ','
ENCLOSED BY '\"'
ESCAPED BY '\''
LINES
TERMINATED BY '\r\n';
```

该语句将把 person 表中所有记录导入到 C 盘目录下的 person1.txt 文本文件中。

FIELDS TERMINATED BY ','表示字段之间用逗号分隔。ENCLOSED BY '\"'表示每个字段用双引号括起来。ESCAPED BY '\''表示将系统默认的转义字符替换为单引号。LINES TERMINATED BY '\r\n'表示每行以回车换行符结尾，保证每一条记录占一行。

执行成功后，在目录 C:\下生成一个 person1.txt 文件，打开文件内容如下：

```
"1","Green","21","Lawyer"
"2","Suse","22","dancer"
"3","Mary","24","Musician"
"4","Willam","20","sports man"
"5","Laura","25",'N'
"6","Evans","27","secretary"
"7","Dale","22","cook"
"8","Edison","28","singer"
"9","Harry","21","magician"
"10","Harriet","19","pianist"
```

可以看到，所有的字段值都被双引号包括；第 5 条记录中空值的表示形式为“N”，即使用单

引号替换了反斜线转义字符。

【例 18.12】使用 SELECT...INTO OUTFILE 将 test 数据库中的 person 表中的记录导出到文本文件，使用 LINES 选项，要求每行记录以字符串“>”开始，以“<end>”字符串结尾，执行的命令如下：

```
SELECT * FROM test.person INTO OUTFILE "C:\person2.txt"
LINES
STARTING BY '> '
TERMINATED BY '<end>';
```

执行成功后，在目录 C:\下生成一个 person2.txt 文件，打开文件内容如下：

```
> 1  Green   21  Lawyer <end>> 2 Suse    22  dancer <end>> 3 Mary    24
 Musician <end>> 4  Willam  20  sports man <end>> 5 Laura   25  \N  <end>>  6
 Evans   27  secretary <end>> 7  Dale    22  cook <end>> 8  Edison  28
 singer <end>> 9 Harry   21  magician <end>> 10  Harriet 19  pianist <end>
```

可以看到，虽然将所有的字段值导出到文本文件中，但是所有的记录没有分行区分，出现这种情况是因为 TERMINATED BY 选项替换了系统默认的“\n”换行符，如果希望换行显示，则需要修改导出语句，输入下面语句：

```
SELECT * FROM test.person INTO OUTFILE "C:\person2.txt"
LINES
STARTING BY '> '
TERMINATED BY '<end>\r\n';
```

执行完语句之后，换行显示每条记录，结果如下：

```
> 1  Green   21  Lawyer <end>
> 2  Suse        22  dancer <end>
> 3  Mary        24  Musician <end>
> 4  Willam  20  sports man <end>
> 5  Laura       25  \N <end>
> 6  Evans   27  secretary <end>
> 7  Dale        22  cook <end>
> 8  Edison  28  singer <end>
> 9  Harry       21  magician <end>
> 10 Harriet 19  pianist <end>
```

18.4.2 使用 MySQLdump 命令导出文本文件

除了使用 SELECT… INTO OUTFILE 语句导出文本文件之外，还可以使用 MySQLdump。本章开始介绍了使用 MySQLdump 备份数据库，该工具不仅可以将数据导出为包含 CREATE、INSERT 的 sql 文件，也可以导出为纯文本文件。

MySQLdump 创建一个包含创建表的 CREATE TABLE 语句的 tablename.sql 文件和一个包含其数据的 tablename.txt 文件。MySQLdump 导出文本文件的基本语法格式如下：

```
mysqldump -T path-u root -p dbname [tables] [OPTIONS]

--OPTIONS 选项
--fields-terminated-by=value
--fields-enclosed-by=value
--fields-optionally-enclosed-by=value
--fields-escaped-by=value
--lines-terminated-by=value
```

只有指定了-T 参数才可以导出纯文本文件；path 表示导出数据的目录；tables 为指定要导出的表名称，如果不指定，将导出数据库 dbname 中所有的表；[OPTIONS]为可选参数选项，这些选项需要结合-T 选项使用。使用 OPTIONS 常见的取值有：

- --fields-terminated-by=value：设置字段之间的分隔字符，可以为单个或多个字符，默认情况下为制表符“\t”。
- --fields-enclosed-by=value：设置字段的包围字符。
- --fields-optionally-enclosed-by=value：设置字段的包围字符，只能为单个字符，只能包括 CHAR 和 VERCHAR 等字符数据字段。
- --fields-escaped-by=value：控制如何写入或读取特殊字符，只能为单个字符，即设置转义字符，默认值为反斜线“\”。
- --lines-terminated-by=value：设置每行数据结尾的字符，可以为单个或多个字符，默认值为“\n”。



与 SELECT…INTO OUTFILE 语句中的 OPTIONS 各个参数设置不同，这里 OPTIONS 各个选项等号后面的 value 值不要用引号括起来。

【例 18.13】使用 MySQLdump 将 test 数据库中的 person 表中的记录导出到文本文件，执行的命令如下：

```
mysqldump -T C:\test person -u root-p
```

语句执行成功，系统 C 盘目录下面将会有两个文件，分别为 person.sql 和 person.txt。person.sql 包含创建 person 表的 CREATE 语句，其内容如下：

```
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `person`
--

DROP TABLE IF EXISTS `person`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `person` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` char(40) NOT NULL DEFAULT '',
  `age` int(11) NOT NULL DEFAULT '0',
  `info` char(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2011-08-19 15:02:16
```

备份文件中的信息如 18.1.1 节介绍。

person.txt 包含数据包中的数据，其内容如下：

```
1    Green      21      Lawyer
2    Suse       22      dancer
3    Mary       24      Musician
4    Willam     20      sports man
5    Laura      25      \N
6    Evans      27      secretary
7    Dale       22      cook
8    Edison     28      singer
9    Harry      21      magician
10   Harriet    19      pianist
```

【例 18.14】使用 MySQLdump 命令将 test 数据库中的 person 表中的记录导出到文本文件，使用 FIELDS 选项，要求字段之间使用逗号“,”间隔，所有字符类型字段值用双引号括起来，定义转义字符为问号“?”，每行记录以回车换行符“\r\n”结尾，执行的命令如下：

```
C:\>mysqldump -T C:\backup test person -u root -p --fields-terminated-by=,
--fields-optionally-enclosed-by=\"                    --fields-escaped-by=?
--lines-terminated-by=\r\n
```

上面语句要在一行中输入，语句执行成功，系统 C:\backup 目录下面将会有两个文件，分别为 person.sql 和 person.txt。person.sql 包含创建 person 表的 CREATE 语句，其内容与前面例子中的相同，person.txt 文件的内容与上一个例子不同，显示如下：

```
1,"Green",21,"Lawyer"
2,"Suse",22,"dancer"
```

```
3,"Mary",24,"Musician"
4,"Willam",20,"sports man"
5,"Laura",25,?N
6,"Evans",27,"secretary"
7,"Dale",22,"cook"
8,"Edison",28,"singer"
9,"Harry",21,"magician"
10,"Harriet",19,"pianist"
```

可以看到，只有字符类型的值被双引号括了起来，而数值类型的值没有；第 5 行记录中的 NULL 值表示为“?N”，使用问号“?”替代了系统默认的反斜线转义字符“\”。

18.4.3 使用 MySQL 命令导出文本文件

MySQL 是一个功能丰富的工具命令，使用 MySQL 还可以在命令行模式下执行 SQL 指令，将查询结果导入到文本文件中。相比 MySQLdump，MySQL 工具导出的结果可读性更强。

如果 MySQL 服务器是单独的机器，用户是在一个 client 上进行操作，用户要把数据结果导入到 client 机器上。可以使用 MySQL -e 语句。

使用 MySQL 导出数据文本文件语句的基本格式如下：

```
mysql -u root -p --execute= "SELECT 语句" dbname > filename.txt
```

该命令使用--execute 选项，表示执行该选项后面的语句并退出，后面的语句必须用双引号括起来，dbname 为要导出的数据库名称；导出的文件中不同列之间使用制表符分隔，第 1 行包含了各个字段的名称。

【例 18.15】使用 MySQL 语句，导出 test 数据库中 person 表中的记录到文本文件，输入语句如下：

```
mysql -u root -p --execute="SELECT * FROM person;" test > C:\person3.txt
```

语句执行完毕之后，系统 C 盘目录下面将会有名称为 person3.txt 的文本文件，其内容如下：

```
id   name       age     info
1    Green      21      Lawyer
2    Suse       22      dancer
3    Mary       24      Musician
4    Willam     20      sports man
5    Laura      25      NULL
6    Evans      27      secretary
7    Dale       22      cook
8    Edison     28      singer
9    Harry      21      magician
10   Harriet    19      pianist
```

可以看到，person3.txt 文件中包含了每个字段的名称和各条记录，该显示格式与 MySQL 命令行下 SELECT 查询结果显示相同。

使用 MySQL 命令还可以指定查询结果的显示格式，如果某行记录字段很多，可能一行不能完

全显示，可以使用--vartical 参数，将每条记录分为多行显示。

【例 18.16】使用 MySQL 命令导出 test 数据库中 person 表中的记录到文本文件，使用--vertical 参数显示结果，输入语句如下：

```
mysql -u root -p --vertical --execute="SELECT * FROM person;" test > C:\person4.txt
```

语句执行之后，C:\person4.txt 文件中的内容如下：

```
*** 1. row ***
  id: 1
name: Green
 age: 21
info: Lawyer
*** 2. row ***
  id: 2
name: Suse
 age: 22
info: dancer
*** 3. row ***
  id: 3
name: Mary
 age: 24
info: Musician
*** 4. row ***
  id: 4
name: Willam
 age: 20
info: sports man
*** 5. row ***
  id: 5
name: Laura
 age: 25
info: NULL
*** 6. row ***
  id: 6
name: Evans
 age: 27
info: secretary
*** 7. row ***
  id: 7
name: Dale
 age: 22
info: cook
*** 8. row ***
  id: 8
```

```
name: Edison
 age: 28
info: singer
*** 9. row ***
  id: 9
name: Harry
 age: 21
info: magician
*** 10. row ***
  id: 10
name: Harriet
 age: 19
info: pianist
```

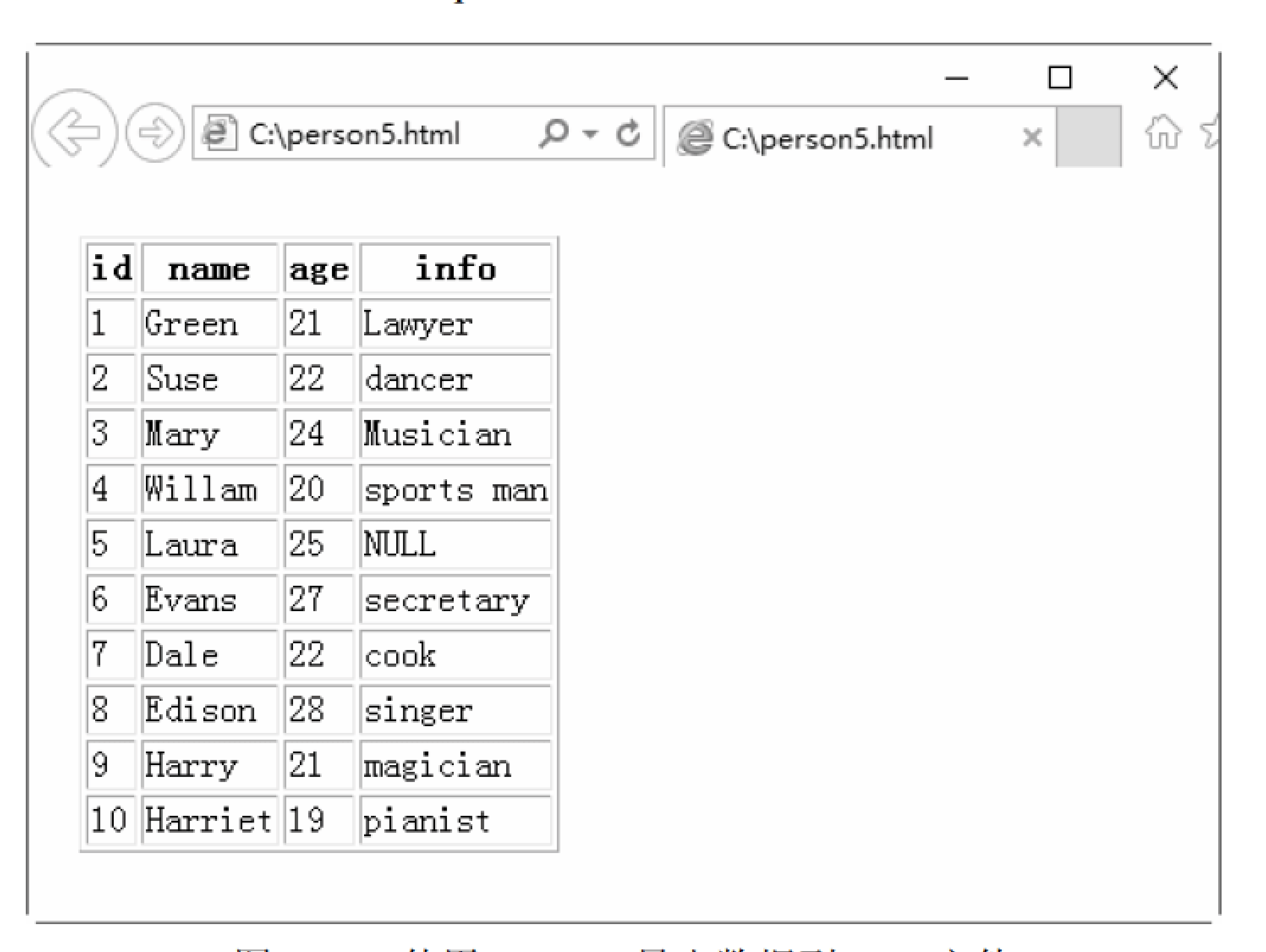
可以看到，SELECT 的查询结果导出到文本文件之后，显示格式发生了变化，如果 person 表中记录内容很长，这样显示将会更加容易阅读。

MySQL 可以将查询结果导出到 html 文件中，使用--html 选项即可。

【例 18.17】使用 MySQL 命令导出 test 数据库中 person 表中的记录到 html 文件，输入语句如下：

```
mysql -u root -p --html --execute="SELECT * FROM person;" test > C:\person5.html
```

语句执行成功，将在 C 盘创建文件 person5.html，该文件在浏览器中显示如图 18-1 所示。



id	name	age	info
1	Green	21	Lawyer
2	Suse	22	dancer
3	Mary	24	Musician
4	Willam	20	sports man
5	Laura	25	NULL
6	Evans	27	secretary
7	Dale	22	cook
8	Edison	28	singer
9	Harry	21	magician
10	Harriet	19	pianist

图 18-1　使用 MySQL 导出数据到 html 文件

如果要将表数据导出到 xml 文件中，可使用--xml 选项。

【例 18.18】使用 MySQL 命令导出 test 数据库中 person 表中的记录到 xml 文件，输入语句如下：

```
mysql -u root -p --xml --execute="SELECT * FROM person;" test > C:\person6.xml
```

语句执行成功，将在 C 盘创建文件 person6.xml，该文件在浏览器中显示如图 18-2 所示。

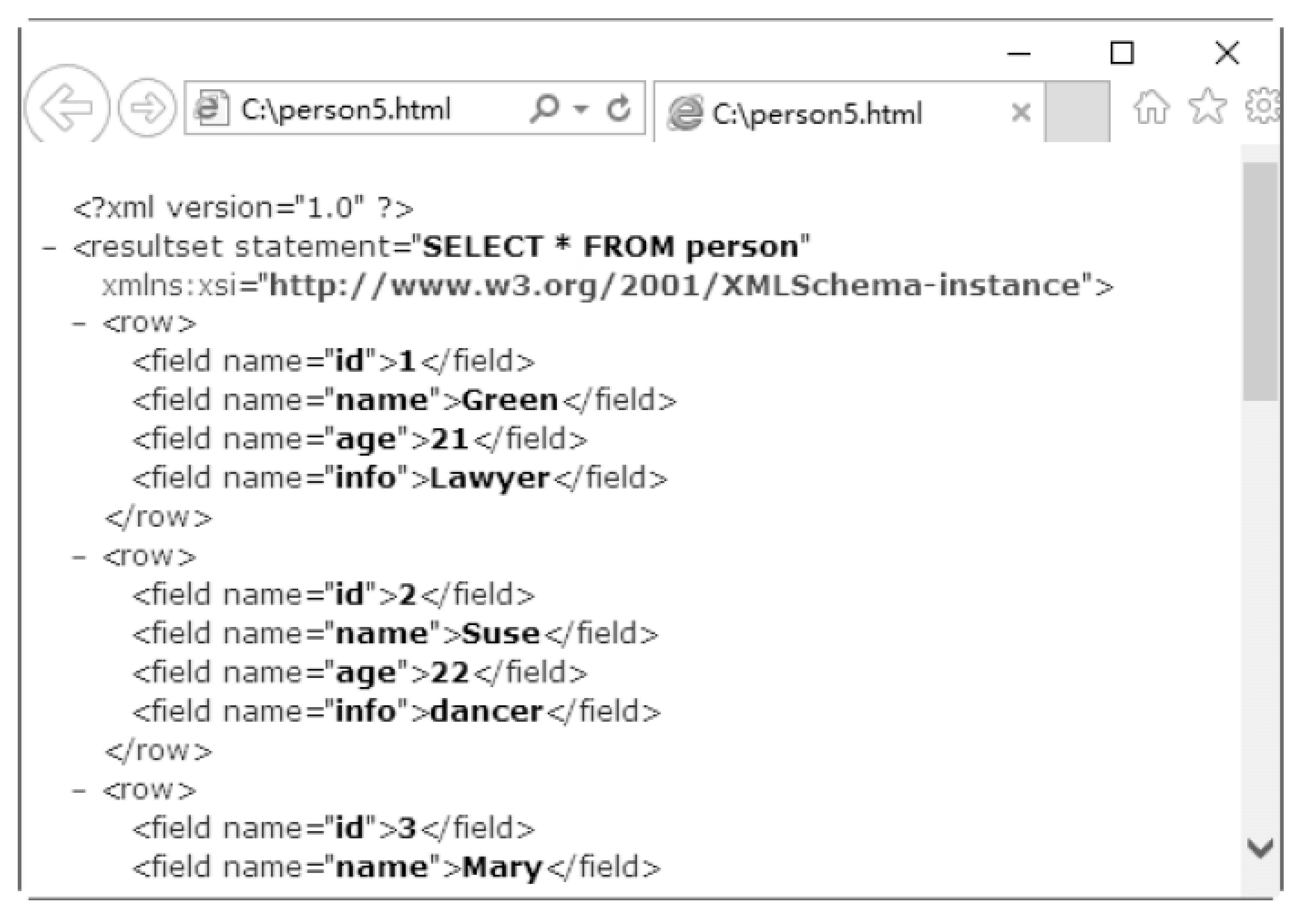


图 18-2　使用 MySQL 导出数据到 xml 文件

18.4.4　使用 LOAD DATA INFILE 方式导入文本文件

MySQL 允许将数据导出到外部文件，也可以从外部文件导入数据。MySQL 提供了一些导入数据的工具，这些工具有 LOAD DATA 语句、source 命令和 MySQL 命令。LOAD DATA INFILE 语句用于高速地从一个文本文件中读取行，并装入一个表中。文件名称必须为文字字符串。本节将介绍 LOAD DATA 语句的用法。

LOAD DATA 语句的基本格式如下：

```
   LOAD DATA  INFILE 'filename.txt' INTO TABLE tablename [OPTIONS] [IGNORE number
LINES]

   -- OPTIONS 选项
      FIELDS  TERMINATED BY 'value'
   FIELDS  [OPTIONALLY] ENCLOSED BY 'value'
   FIELDS  ESCAPED BY 'value'
   LINES  STARTING BY 'value'
   LINES  TERMINATED BY 'value'
```

可以看到 LOAD DATA 语句中，关键字 INFILE 后面的 filename 文件为导入数据的来源；tablename 表示待导入的数据表名称；[OPTIONS]为可选参数选项，OPTIONS 部分的语法包括 FIELDS 和 LINES 子句，其可能的取值有：

- FIELDS　TERMINATED BY 'value'：设置字段之间的分隔字符，可以为单个或多个字符，默认情况下为制表符“\t”。
- FIELDS　[OPTIONALLY] ENCLOSED BY 'value'：设置字段的包围字符，只能为单个字符。如果使用了 OPTIONALLY，则只有 CHAR 和 VERCHAR 等字符数据字段被包括。
- FIELDS　ESCAPED BY 'value'：控制如何写入或读取特殊字符，只能为单个字符，即设置转义字符，默认值为“\”。

- LINES STARTING BY 'value'：设置每行数据开头的字符，可以为单个或多个字符，默认情况下不使用任何字符。
- LINES TERMINATED BY 'value'：设置每行数据结尾的字符，可以为单个或多个字符，默认值为“\n”。

IGNORE number LINES 选项表示忽略文件开始处的行数，number 表示忽略的行数。执行 LOAD DATA 语句需要 FILE 权限。

【例 18.19】使用 LOAD DATA 命令将 C:\person0.txt 文件中的数据导入到 test 数据库中的 person 表，输入语句如下：

```
LOAD DATA  INFILE 'C:\person0.txt' INTO TABLE test.person;
```

恢复之前，将 person 表中的数据全部删除，登录 MySQL，使用 DELETE 语句，语句如下：

```
mysql> USE test;
Database changed;
mysql> DELETE FROM person;
Query OK, 10 rows affected (0.00 sec)
```

从 person0.txt 文件中恢复数据，语句如下：

```
mysql> LOAD DATA  INFILE 'C:\person0.txt' INTO TABLE test.person;
Query OK, 10 rows affected (0.00 sec)
Records: 10  Deleted: 0  Skipped: 0  Warnings: 0

mysql> SELECT * FROM person;
+----+---------+-----+------------+
| id | name    | age | info       |
+----+---------+-----+------------+
|  1 | Green   |  21 | Lawyer     |
|  2 | Suse    |  22 | dancer     |
|  3 | Mary    |  24 | Musician   |
|  4 | Willam  |  20 | sports man |
|  5 | Laura   |  25 | NULL       |
|  6 | Evans   |  27 | secretary  |
|  7 | Dale    |  22 | cook       |
|  8 | Edison  |  28 | singer     |
|  9 | Harry   |  21 | magician   |
| 10 | Harriet |  19 | pianist    |
+----+---------+------+------------+
10 rows in set (0.00 sec)
```

可以看到，语句执行成功之后，原来的数据重新恢复到了 person 表中。

【例 18.20】使用 LOAD DATA 命令将 C:\person1.txt 文件中的数据导入到 test 数据库中的 person 表，使用 FIELDS 选项和 LINES 选项，要求字段之间使用逗号“,”间隔，所有字段值用双引号括起来，定义转义字符为单引号“\'”，每行记录以回车换行符“\r\n”结尾，输入语句如下：

```
LOAD DATA INFILE 'C:\person1.txt' INTO TABLE test.person
FIELDS
TERMINATED BY ','
ENCLOSED BY '\"'
ESCAPED BY '\''
LINES
TERMINATED BY '\r\n';
```

恢复之前，将 person 表中的数据全部删除，使用 DELETE 语句，执行过程如下：

```
mysql> DELETE FROM person;
Query OK, 10 rows affected (0.00 sec)
```

从 person1.txt 文件中恢复数据，执行过程如下：

```
mysql> LOAD DATA  INFILE 'C:\person1.txt' INTO TABLE test.person
    -> FIELDS
    -> TERMINATED BY ','
    -> ENCLOSED BY '\"'
    -> ESCAPED BY '\''
    -> LINES
    -> TERMINATED BY '\r\n';
Query OK, 10 rows affected (0.00 sec)
Records: 10  Deleted: 0  Skipped: 0  Warnings: 0
```

语句执行成功，使用 SELECT 语句查看 person 表中的记录，结果与前一个例子相同。

18.4.5 使用 MySQLimport 命令导入文本文件

使用 MySQLimport 可以导入文本文件，并且不需要登录 MySQL 客户端。MySQLimport 命令提供许多与 LOAD DATA INFILE 语句相同的功能，大多数选项直接对应 LOAD DATA INFILE 子句。使用 MySQLimport 语句需要指定所需的选项、导入的数据库名称以及导入的数据文件的路径和名称。MySQLimport 命令的基本语法格式如下：

```
mysqlimport -u root-p dbname filename.txt [OPTIONS]

--OPTIONS 选项
--fields-terminated-by=value
--fields-enclosed-by=value
--fields-optionally-enclosed-by=value
--fields-escaped-by=value
--lines-terminated-by=value
--ignore-lines=n
```

dbname 为导入的表所在的数据库名称。注意，MySQLimport 命令不指定导入数据库的表名称，数据表的名称由导入文件名称确定，即文件名作为表名，导入数据之前该表必须存在。[OPTIONS] 为可选参数选项，其常见的取值有：

- --fields-terminated-by= 'value'：设置字段之间的分隔字符，可以为单个或多个字符，默认情况下为制表符“\t”。
- --fields-enclosed-by= 'value'：设置字段的包围字符。
- --fields-optionally-enclosed-by= 'value'：设置字段的包围字符，只能为单个字符，包括 CHAR 和 VERCHAR 等字符数据字段。
- --fields-escaped-by= 'value'：控制如何写入或读取特殊字符，只能为单个字符，即设置转义字符，默认值为反斜线“\”。
- --lines-terminated-by= 'value'：设置每行数据结尾的字符，可以为单个或多个字符，默认值为“\n”。
- --ignore-lines=n：忽视数据文件的前 n 行。

【例 18.21】使用 MySQLimport 命令将 C:\backup 目录下的 person.txt 文件内容导入到 test 数据库中，字段之间使用逗号“,”间隔，字符类型字段值用双引号括起来，将转义字符定义为问号“?”，每行记录以回车换行符“\r\n”结尾，执行的命令如下：

```
  C:\ >mysqlimport -u root -p test C:\backup/person.txt --fields-terminated-by=,
--fields-optionally-enclosed-by=\" --fields-escaped-by=? --lines-terminated
-by=\r\n
```

上面语句要在一行中输入，语句执行成功，将把 person.txt 中的数据导入到数据库。

除了前面介绍的几个选项之外，MySQLimport 支持许多选项，常见的选项有：

- --columns=column_list, -c column_list：该选项采用逗号分隔的列名作为其值。列名的顺序指示如何匹配数据文件列和表列。
- --compress，-C：压缩在客户端和服务器之间发送的所有信息（如果二者均支持压缩）。
- -d，--delete：导入文本文件前清空表。
- --force，-f：忽视错误。例如，如果某个文本文件的表不存在，继续处理其他文件。不使用--force，如果表不存在，则 MySQLimport 退出。
- --host=host_name，-h host_name：将数据导入给定主机上的 MySQL 服务器。默认主机是 localhost。
- --ignore，-i：参见--replace 选项的描述。
- --ignore-lines=n：忽视数据文件的前 n 行。
- --local，-L：从本地客户端读入输入文件。
- --lock-tables，-l：处理文本文件前锁定所有表以便写入。这样可以确保所有表在服务器上保持同步。
- --password[=password]，-p[password]：当连接服务器时使用的密码。如果使用短选项形式（-p），选项和密码之间不能有空格。如果在命令行中--password 或-p 选项后面没有密码值，则提示输入一个密码。
- --port=port_num，-P port_num：用于连接的 TCP/IP 端口号。
- --protocol={TCP | SOCKET | PIPE | MEMORY}：使用的连接协议。
- --replace，-r --replace 和--ignore 选项控制复制唯一键值已有记录的输入记录的处理。如果指定--replace，新行替换有相同的唯一键值的已有行；如果指定--ignore，复制已有的唯一键值的输入行被跳过；如果不指定这两个选项，当发现一个复制键值时会出现一个错误，

并且忽视文本文件的剩余部分。

- --silent，-s：沉默模式。只有出现错误时才输出信息。
- --user=user_name，-u user_name：当连接服务器时 MySQL 使用的用户名。
- --verbose，-v：冗长模式。打印出程序操作的详细信息。
- --version，-V：显示版本信息并退出。

18.5　实战演练——数据的备份与恢复

备份有助于保护数据库，通过备份可以完整保存 MySQL 中各个数据库的特定状态。在系统出现故障、数据丢失或者不合理操作对数据库造成灾难时，可以通过恢复恢复数据库中的数据。作为 MySQL 的管理人员，应该定期地备份所有活动的数据库，以免发生数据丢失。因此，无论怎样强调数据库的备份工作都不过分。本章综合案例将向读者提供数据库备份与恢复的方法与过程。

1. 案例目的

按照操作过程完成对 test 数据库的备份和恢复。

01 使用 MySQLdump 命令将 suppliers 表备份到文件 C:\bktestdir\suppliers_bk.sql。
02 使用 MySQL 命令恢复 suppliers 表到 test 数据库中。
03 使用 SELECT… INTO OUTFILE 语句导出 suppliers 表中的记录，导出文件位于目录 C:\bktestdir 下，名称为 suppliers_out.txt。
04 使用 LOAD DATA INFILE 语句导入 suppliers_out.txt 数据到 suppliers 表。
05 使用 MySQLdump 命令将 suppliers 表中的记录导出到文件 C:\bktestdir\suppliers_html.html。

2. 案例操作过程

01 使用 MySQLdump 命令将 suppliers 表备份到文件 C:\bktestdir\suppliers_bk.sql。

首先创建系统目录，在系统 C 盘下面新建文件夹 bktestdir，然后打开命令行窗口，输入语句如下：

```
C:\ >mysqldump -u root -p test suppliers > C:\bktestdir\suppliers_bk.sql
Enter password: **
```

语句执行完毕，打开目录 C:\bktestdir，可以看到已经创建好的备份文件 suppliers_bk.sql，内容如下：

```
-- MySQL dump 10.13  Distrib 5.7.10, for Win32 (x86)
--
-- Host: localhost    Database: test
-- ------------------------------------------------------
-- Server version   5.7.10

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
```

```
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_
CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `suppliers`
--

DROP TABLE IF EXISTS `suppliers`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `suppliers` (
  `s_id` int(11) NOT NULL AUTO_INCREMENT,
  `s_name` char(50) NOT NULL,
  `s_city` char(50) DEFAULT NULL,
  `s_zip` char(10) DEFAULT NULL,
  `s_call` char(50) NOT NULL,
  PRIMARY KEY (`s_id`)
) ENGINE=InnoDB AUTO_INCREMENT=108 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `suppliers`
--

LOCK TABLES `suppliers` WRITE;
/*!40000 ALTER TABLE `suppliers` DISABLE KEYS */;
INSERT INTO `suppliers` VALUES (101,'FastFruit Inc.','Tianjin','463400',
'48075'),(102,'LT Supplies',' Chongqing','100023','44333'),(103,'ACME',
'Shanghai','100024','90046'),(104,'FNK Inc.','Zhongshan','212021','11111'),
 (105,'Good Set','Taiyuang','230009','22222'),(106,'Just Eat Ours','Beijing',
'010','45678'),(107,'DK Inc.','Qingdao',  '230009','33332');
/*!40000 ALTER TABLE `suppliers` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

02 使用 MySQL 命令将备份文件 suppliers_bk.sql 中的数据恢复 suppliers 表。

为了验证恢复之后数据的正确性，删除 suppliers 表中的所有记录，登录 MySQL，输入语句：

```
mysql> USE test;
Database changed
```

```
mysql> DELETE FROM suppliers;
Query OK, 7 rows affected (0.00 sec)
```

此时，suppliers 表中不再有任何数据记录，在 MySQL 命令行输入恢复语句如下：

```
mysql> source C:\bktestdir\suppliers_bk.sql;
```

语句执行过程中会出现多行提示信息，执行成功之后使用 SELECT 语句查询 suppliers 表，内容如下：

```
mysql> SELECT * FROM suppliers;
+------+----------------+-------------+---------+--------+
| s_id | s_name       | s_city     | s_zip  | s_call |
+------+----------------+-------------+---------+--------+
|  101 | FastFruit Inc. | Tianjin    | 463400 | 48075|
|  102 | LT Supplies  | Chongqing | 100023 | 44333|
|  103 | ACME       | Shanghai  | 100024 | 90046|
|  104 | FNK Inc.    | Zhongshan | 212021 | 11111|
|  105 | Good Set    | Taiyuang  | 230009 | 22222|
|  106 | Just Eat Ours | Beijing    | 010    | 45678 |
|  107 | DK Inc.     | Qingdao   | 230009 | 33332 |
+------+----------------+-------------+----------+--------+
7 rows in set (0.00 sec)
```

由查询结果可以看到，恢复操作成功。

03 使用 SELECT… INTO OUTFILE 语句导出 suppliers 表中的记录，导出文件位于目录 C:\bktestdir 下，名称为 suppliers_out.txt。

执行过程如下：

```
mysql>     SELECT     *     FROM     test.suppliers     INTO     OUTFILE
"C:\bktestdir\suppliers_out.txt"
     -> FIELDS
     -> TERMINATED BY ','
     -> ENCLOSED BY '\"'
     -> LINES
     -> STARTING BY '<'
     -> TERMINATED BY '>\r\n';
Query OK, 7 rows affected (0.00 sec)
```

TERMINATED BY ','指定不同字段之间使用逗号分隔开；ENCLOSED BY '\"'指定字段值使用双引号包括；STARTING BY '< '指定每行记录以左箭头符号开始；TERMINATED BY '>\r\n';指定每行记录以右箭头符号和回车换行符结束。语句执行完毕，打开目录 C:\bktestdir，可以看到已经创建好的导出文件 suppliers_out.txt，内容如下：

```
<"101","FastFruit Inc.","Tianjin","463400","48075">
<"102","LT Supplies","Chongqing","100023","44333">
<"103","ACME","Shanghai","100024","90046">
<"104","FNK Inc.","Zhongshan","212021","11111">
<"105","Good Set","Taiyuang","230009","22222">
<"106","Just Eat Ours","Beijing","010","45678">
<"107","DK Inc.","Qingdao","230009","33332">
```

04 使用 LOAD DATA INFILE 语句导入 suppliers_out.txt 数据到 suppliers 表。

首先使用 DELETE 语句删除 suppliers 表中的所有记录，然后输入导入语句：

```
mysql> LOAD DATA INFILE 'C:\bktestdir\suppliers_out.txt' INTO TABLE test.suppliers
    -> FIELDS
    -> TERMINATED BY ','
    -> ENCLOSED BY '\"'
    -> LINES
    -> STARTING BY '<'
    -> TERMINATED BY '>\r\n';
Query OK, 7 rows affected (0.00 sec)
Records: 7  Deleted: 0  Skipped: 0  Warnings: 0
```

语句执行之后，suppliers_out.txt 文件中的数据将导入 suppliers 表中，由于导出 txt 文件时指定了一些特殊字符，因此恢复语句中也要指定这些字符，已确保恢复后数据的完整性和正确性。

05 使用 MySQLdump 命令将 suppliers 表中的记录导出到文件 C:\bktestdir\suppliers_html.html。

导出表数据到 html 文件，使用 MySQL 命令时需要指定--html 选项，在 Windows 命令行窗口输入导出语句如下：

```
mysql -u root -p --html --execute="SELECT * FROM suppliers;" test > C:\bktestdir\suppliers_html.html
```

语句执行完毕，打开目录 C:\bktestdir，可以看到已经创建好的导出文件 suppliers_html.html，读者可以使用浏览器打开该文件，在浏览器中显示格式和内容如表 18-1 所示。

表 18-1 浏览器中显示导出文件的内容

s_id	s_name	s_city	s_zip	s_call
101	FastFruit Inc.	Tianjin	463400	48075
102	LT Supplies	Chongqing	100023	44333
103	ACME	Shanghai	100024	90046
104	FNK Inc.	Zhongshan	212021	11111
105	Good Set	Taiyuang	230009	22222
106	Just Eat Ours	Beijing	010	45678
107	DK Inc.	Qingdao	230009	33332

18.6 高手私房菜

技巧 1：MySQLdump 备份的文件只能在 MySQL 中使用吗？

MySQLdump 备份的文本文件实际是数据库的一个副本，使用该文件不仅可以在 MySQL 中恢复数据库，而且通过对该文件的简单修改，可以使用该文件在 SQL Server 或者 Sybase 等其他数据库中恢复数据库。这在某种程度上实现了数据库之间的迁移。

技巧 2：如何选择备份工具？

直接复制数据文件是最为直接、快速的备份方法，但缺点是基本上不能实现增量备份。备份时必须确保没有使用这些表。如果在复制一个表的同时服务器正在修改它，则复制无效。备份文件时，最好关闭服务器，然后重新启动服务器。为了保证数据的一致性，需要在备份文件前，执行以下 SQL 语句：

```
FLUSH TABLES WITH READ LOCK;
```

也就是把内存中的数据都刷新到磁盘中，同时锁定数据表，以保证复制过程中不会有新的数据写入。这种方法备份出来的数据恢复也很简单，直接复制回原来的数据库目录下即可。

MySQLhotcopy 是一个 PERL 程序，它使用 LOCK TABLES、FLUSH TABLES 和 cp 或 scp 来快速备份数据库。它是备份数据库或单个表的最快的途径，但它只能运行在数据库文件所在的机器上，并且 MySQLhotcopy 只能用于备份 MyISAM 表。MySQLhotcopy 适合于小型数据库的备份，数据量不大，可以使用 MySQLhotcopy 程序每天进行一次完全备份。

MySQLdump 将数据表导成 SQL 脚本文件，在不同的 MySQL 版本之间升级时相对比较合适，这也是最常用的备份方法。MySQLdump 比直接复制要慢些。

技巧 3：使用 MySQLdump 备份整个数据库成功，把表和数据库都删除了，但使用备份文件却不能恢复数据库？

出现这种情况，是因为备份的时候没有指定--databases 参数。默认情况下，如果只指定数据库名称，MySQLdump 备份的是数据库中所有的表，而不包括数据库的创建语句，例如：

```
mysqldump -u root -p booksDB > c:\backup\booksDB_20160101.sql
```

该语句只备份了 booksDB 数据库下所有的表，读者打开该文件，可以看到文件中不包含创建 booksDB 数据库的 CREATE DATABASE 语句，因此如果把 booksDB 也删除了，使用该 sql 文件不能恢复以前的表，恢复时会出现 ERROR 1046 (3D000): No database selected 的错误信息。必须在 MySQL 命令行下创建 booksDB 数据库，并使用 use 语句选择 booksDB 之后才可以恢复。而下面的语句，数据库删除之后，可以正常恢复备份时的状态。

```
mysqldump -u root -p --databases  booksDB > C:\backup\books_DB_20160101.sql
```

该语句不仅备份了所有数据库下的表结构，而且包括创建数据库的语句。

18.7　经典习题

（1）同时备份 test 数据库中的 fruits 和 suppliers 表，然后删除两个表中的内容并恢复。

（2）将 test 数据库中不同的数据表中的数据，导出到 xml 文件或者 html 文件，并查看文件内容。

（3）使用 MySQL 命令导出 fruits 表中的记录，并将查询结果以垂直方式显示写入导出文件。

第 19 章　PHP 操作 MySQL 数据库

PHP 是一种简单的、面向对象的、解释型的、健壮的、安全的、性能非常高的、独立于架构的、可移植的和动态的脚本语言。而 MySQL 是快速和开源的网络数据库系统。PHP 和 MySQL 的结合是目前 Web 开发的黄金组合，那么 PHP 是如何操作 MySQL 数据库的呢？本章将开始学习使用 PHP 操作 MySQL 数据库的各种函数和技巧。

本章学习目标

- 熟悉 PHP 访问 MySQL 数据库的一般步骤
- 熟悉数据库连接前的准备方法
- 掌握 PHP 操作 MySQL 数据库的基本操作
- 掌握添加动态用户信息的方法
- 掌握查询数据信息的方法

19.1　PHP 访问 MySQL 数据库的一般步骤

通过 Web 访问数据库的工作过程。一般分为以下几个步骤。

（1）用户使用浏览器对某个页面发出 HTTP 请求。

（2）服务器端接收到请求，并发送给 PHP 程序进行处理。

（3）PHP 解析代码。在代码中有连接 MySQL 数据库命令和请求特定数据库的某些特定数据的 SQL 命令。根据这些代码，PHP 打开一个和 MySQL 的连接，并且发送 SQL 命令到 MySQL 数据库。

（4）MySQL 接收到 SQL 语句之后，加以执行。执行完毕后返回执行结果到 PHP 程序。

（5）PHP 执行代码，并根据 MySQL 返回的请求结果数据，生成特定格式的 HTML 文件，且传递给浏览器。HTML 经过浏览器渲染，就是用户请求的展示结果。

19.2　连接数据库前的准备工作

默认情况下，从 PHP 5 开始，PHP 不再自动开启对 MySQL 的支持，而是放到扩展函数库中。所以用户需要在扩展函数库中开启 MySQL 函数库。

首先打开 php.ini，找到“；extension=php_mysqli.dll”，去掉该语句前的分号“；”，如图 19-1 所示，保存 php.ini 文件，重新启动 IIS 或 APACHE 服务器即可。

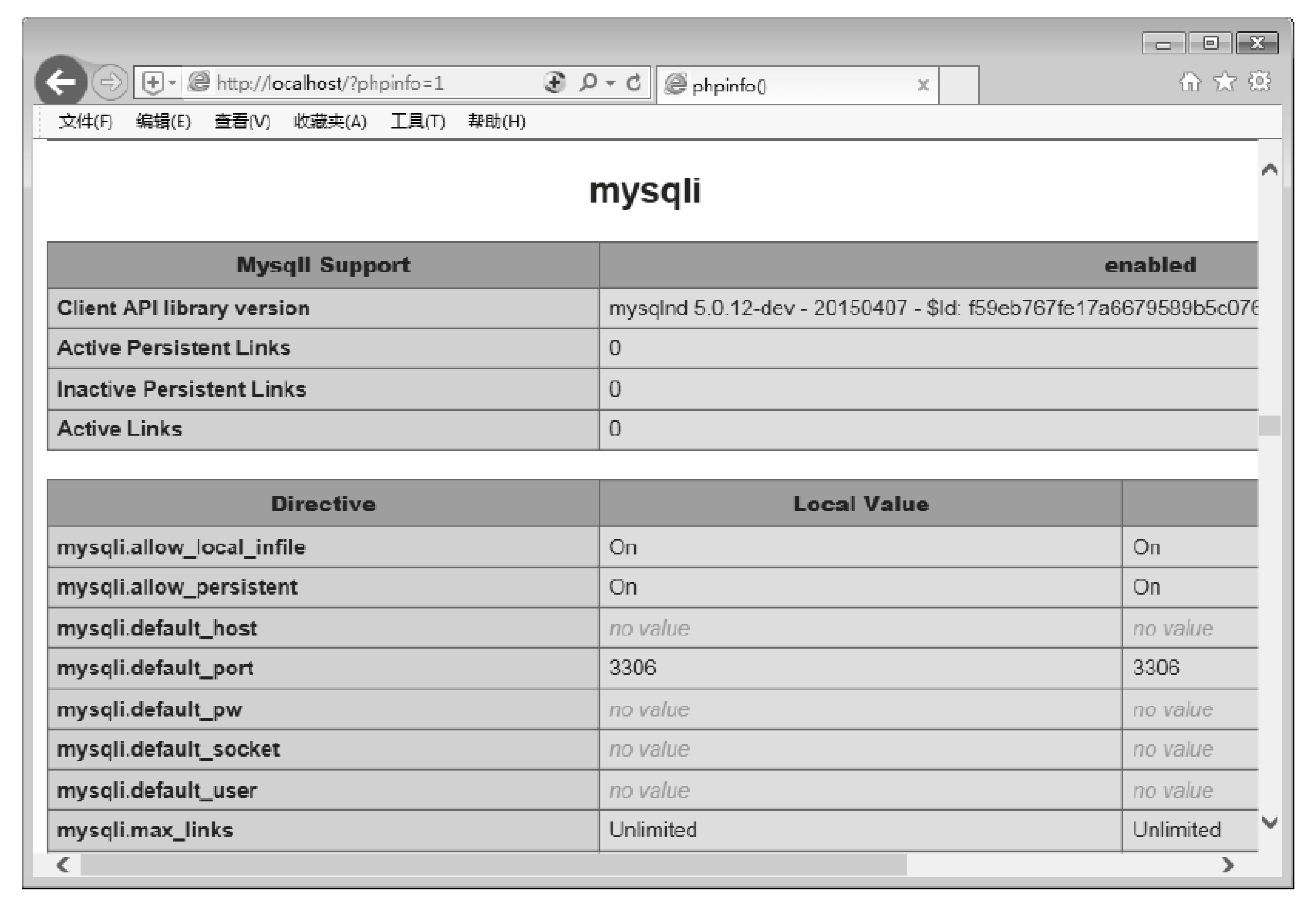
```
php.ini - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
;extension=php_ldap.dll
;extension=php_mbstring.dll
;extension=php_exif.dll      ; Must be after mbstring as it depends on it
extension=php_mysqli.dll
;extension=php_oci8_12c.dll  ; Use with Oracle Database 12c Instant Client
;extension=php_openssl.dll
;extension=php_pdo_firebird.dll
;extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_odbc.dll
;extension=php_pdo_pgsql.dll
;extension=php_pdo_sqlite.dll
;extension=php_pgsql.dll
;extension=php_shmop.dll

; The MIBS data available in the PHP distribution must be installed.
; See http://www.php.net/manual/en/snmp.installation.php
;extension=php_snmp.dll

;extension=php_soap.dll
;extension=php_sockets.dll
```

图 19-1　修改 PHP.ini 文件

配置文件设置完成后，可以通过 phpinfo()函数来检查是否配置成功，如果显示出的 PHP 的环境配置信息中有 mysql 的项目，表示已经开启了对 MySQL 数据库的支持，如图 19-2 所示。



MysqlI Support	enabled
Client API library version	mysqlnd 5.0.12-dev - 20150407 - $Id: f59eb767fe17a6679589b5c076
Active Persistent Links	0
Inactive Persistent Links	0
Active Links	0

Directive	Local Value	
mysqli.allow_local_infile	On	On
mysqli.allow_persistent	On	On
mysqli.default_host	no value	no value
mysqli.default_port	3306	3306
mysqli.default_pw	no value	no value
mysqli.default_socket	no value	no value
mysqli.default_user	no value	no value
mysqli.max_links	Unlimited	Unlimited

图 19-2　PHP 的环境配置页面

19.3　访问数据库

PHP 和 MySQL 数据库是开发动态网站的黄金搭档，本节将讲述 PHP 如何访问 MySQL 数据库。

19.3.1 使用 mysqli_connect()函数连接 MySQL 服务器

PHP 使用 mysqli_connect()函数连接到 MySQL 数据库。

mysqli_connect()函数的格式如下：

```
mysqli_connect('MYSQL 服务器地址', '用户名', '用户密码', '要连接的数据库名')
```

【例 19.1】（实例文件：ch19\19.1.php）

```
<?php
   $db=mysqli_connect('localhost','root','753951','adatabase'); //连接数据库
?>
```

该语句就是通过此函数连接到 MySQL 数据库并且把此连接生成的对象传递给名为$db 的变量，也就是对象$db。其中“MySQL 服务器地址”为 localhost，“用户名”为 root，“用户密码”为本环境 root 设定密码 753951，“要连接的数据库名”为 adatabase。

默认情况下，MySQL 服务的端口号为 3360，如果采用默认的端口号，可以不用指定，如果采用了其他的端口号，比如采用 1066 端口，则需要特别指定，例如 127.0.0.1:1066，表示 MySQL 服务于本地机器的 1066 端口。


提 示

其中 localhost 换成本地地址或者 127.0.0.1，都能实现同样的效果。

如果数据库连接失败，PHP 会发出警告信息，如图 19-3 所示。

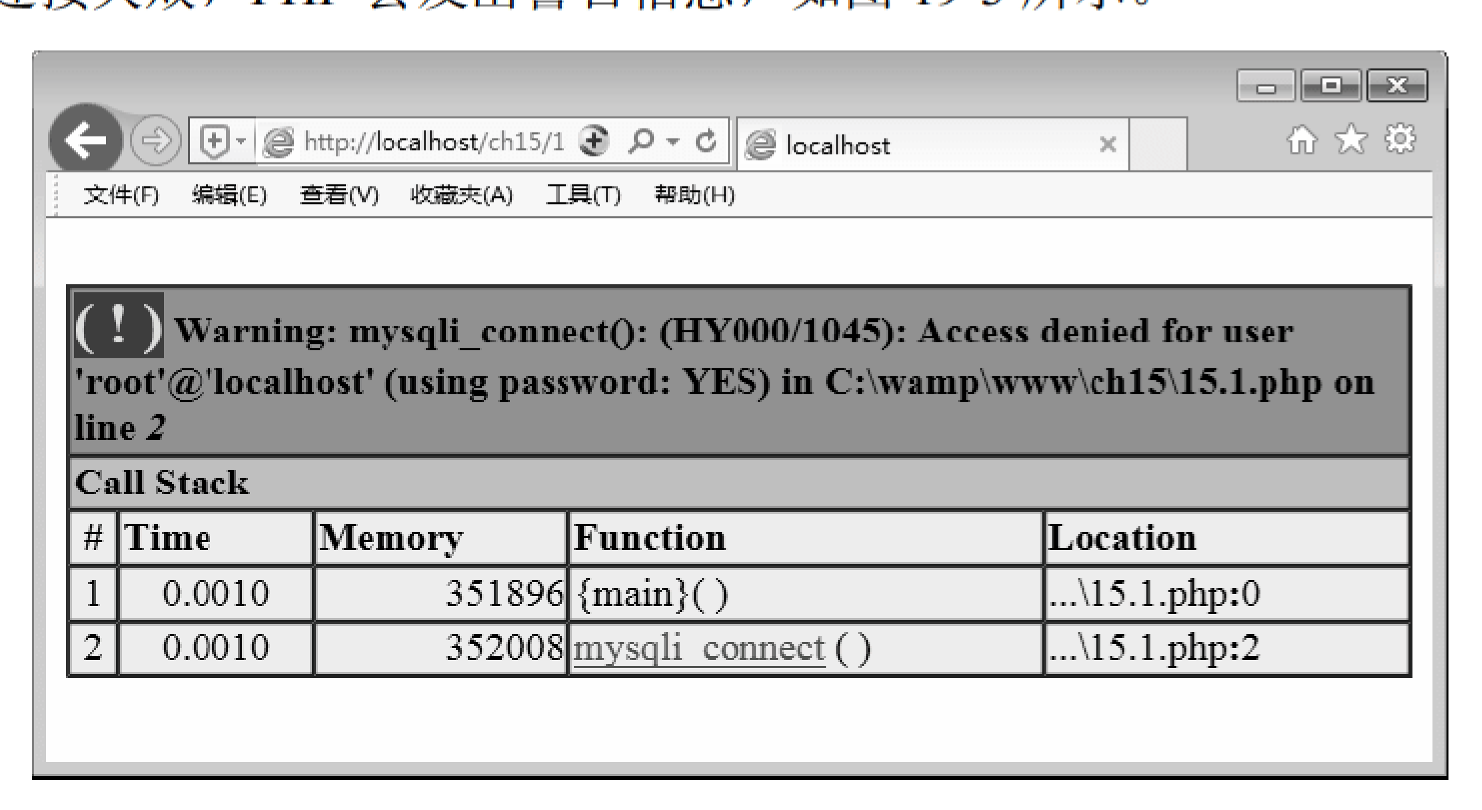


图 19-3 警告信息

警告信息中，提示用 root 账号无法连接到数据库服务器，并且该警告并不会停止脚本的继续执行。可见，这样的提示信息会暴露数据库连接的敏感问题，不利于数据库的安全性。如果想提高安全性，避免错误信息的输出，可以加上@屏蔽错误信息，然后加上 die()函数进行屏蔽的错误处理机制。

【例 19.2】（实例文件：ch19\19.2.php）

```
<?php
```

```
    $db=@mysqli_connect('localhost','root','666666','adatabase')
    or die("无法连接到服务器"); //连接数据库
    print("成功连接到服务器");
    mysqli_close($db);
?>
```

如果数据库连接失败，PHP 会发出警告信息，如图 19-4 所示。这是安全地连接 MySQL 数据库服务器的方法。

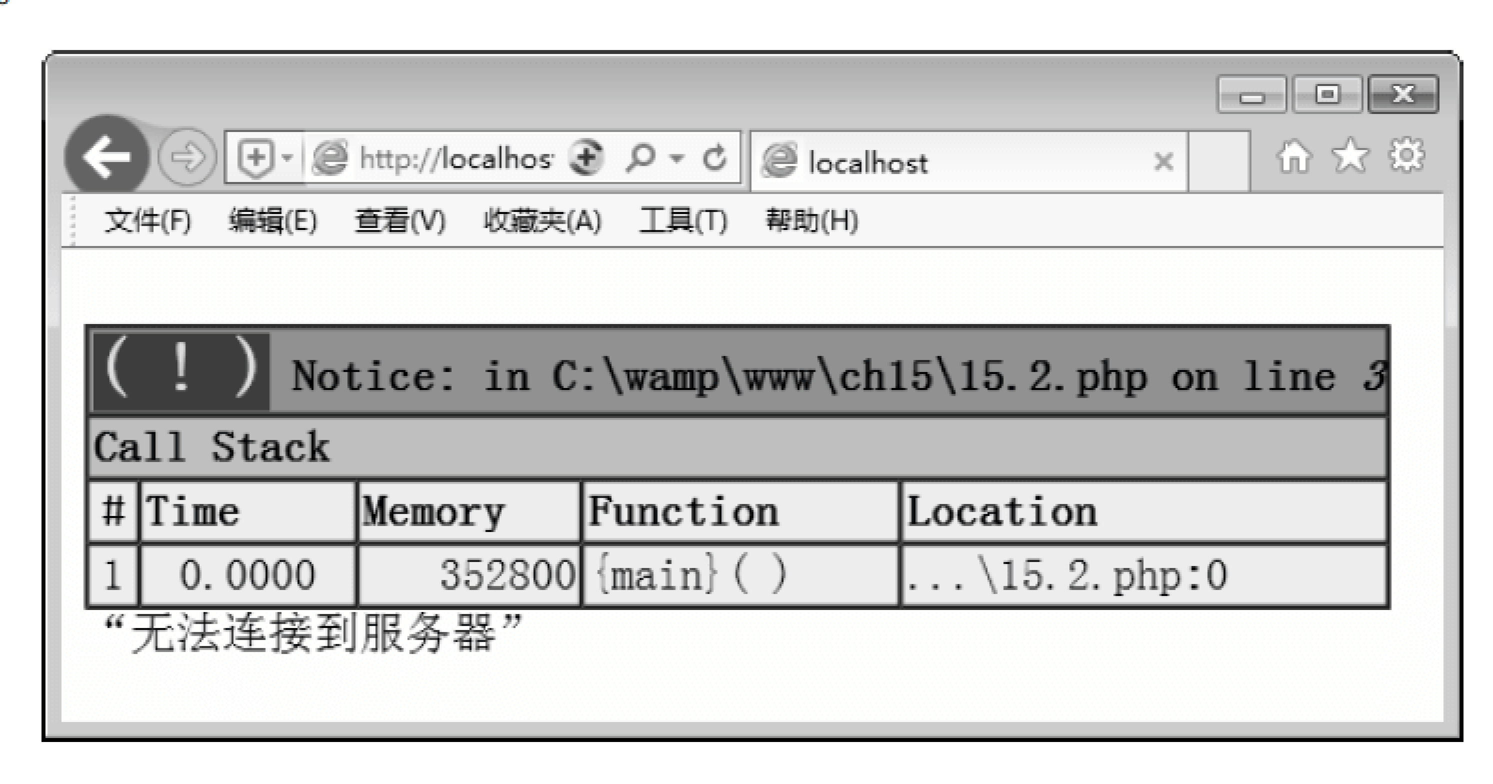


图 19-4 警告信息

19.3.2 使用 mysqli_select_db()函数更改默认的数据库

连接到数据库以后，如果需要更改默认的数据库，需要使用函数 mysqli_select_db()。它的格式为：

```
mysqli_select_db(数据库服务器连接对象，更改后的数据库名)
```

在 19.3.1 小节实例中的$db = mysqli_connect('localhost','root','753951','adatabase');语句已经通过传递参数值 adatabase 确定了需要操作的默认数据库。如果不传递此参数，mysqli_connect()函数只提供“MySQL 服务器地址”、“用户名”和“用户密码”，一样可以连接到 MySQL 数据库服务器并且以相应的用户登录。如果上例的语句变为$db = mysqli_connect('localhost','root','753951');一样是可以成立的。

但是，在这样的情况下，就必须继续选择具体的数据库来进行操作。

如果把 19.1.php 文件中的语句：

```
$db = mysqli_connect('localhost','root','753951','adatabase');
```

修改为以下两个语句替代：

```
$db = mysqli_connect('localhost','root','753951');
mysqli_select_db($db,'adatabase');
```

程序运行效果将完全一样。

在新的语句中 mysqli_select_db($db,'adatabase');语句确定了“数据库服务器连接对象”为$db，“目标数据库名”为 adatabase。

19.3.3 使用 mysqli_close()函数关闭 MySQL 连接

在连接数据库时，可以使用 mysqli_connect()函数。与之相对应，在完成了一次对服务器的使用的情况下，需要关闭此连接，以免对 MySQL 服务器中数据进行误操作并对资源进行释放。一个服务器的连接也是一个对象型的数据类型。

mysqli_close()函数的格式为：

```
mysqli_close(需要关闭的数据库连接对象)
```

在本实例的程序中 mysqli_close($db);语句关闭了“需要关闭的数据库连接对象”为$db 对象。

19.3.4 使用 mysqli_query()函数执行 SQL 语句

使用 mysqli_query()函数执行 SQL 语句，需要向此函数中传递两个参数，一个是 MySQL 数据库服务器连接对象；另一个是以字符串表示的 SQL 语句。mysqli_query()函数的格式如下：

```
mysqli_query(数据库服务器连接对象,SQL 语句)
```

在运行本实例前，用户可以参照前面章节的知识，在 MySQL 服务器上创建 adatabase 数据库，添加数据表 user，数据表 user 主要包括 Id（工号）、Name（姓名）、Age（年龄）、Gender（性别）和 Info（个人信息）字段，然后添加一些演示数据即可。

【例 19.3】（实例文件：ch19\19.3.php）

```
<?php
$db=@mysqli_connect('localhost','root','753951','adatabase')
or die("无法连接到服务器"); //连接数据库
//执行插入数据操作
$sq = "insert into user(Id,Name,Age,Gender,Info)
  values(4,'fangfang',18,'female','She is a 18 years lady')";
$result = mysqli_query($db,$sq); //$result 为 boolean 类型
if ($result) {
 echo "插入数据成功! <br/>";
} else {
 echo "插入数据失败! <br/>";
}
// 执行更新数据操作
$sq = "update user set Name='张芳' where Name='fangfang'";
$result = mysqli_query($db,$sq);
if($result) {
 echo "更新数据成功!<br/>";
} else {
 echo "更新数据失败!<br/>";
}
// 执行查询数据操作
$sq = "select * from user";
$result = mysqli_query($db,$sq); //如果查询成功，$result 为资源类型，保存查询结果集
```

```
mysqli_close($db);
?>
```

程序执行后的结果如图 19-5 所示。



图 19-5　程序运行结果

可见，mysqli_query()函数执行 SQL 语句之后会把结果返回。上例中就是返回结果并且赋值给 $result 变量。

19.3.5　获取查询结果集中的记录数

使用 mysqli_num_rows()函数获取查询结果包含的数据记录的条数，只需要给出返回的数据对象即可。语法格式如下：

```
mysqli_num_rows(result);
```

其中 result 是查询结果对象，此函数只对 select 语句有效。

如果想获取查询、插入、更新和删除操作所影响的行数，需要使用 mysqli_affected_rows 函数。mysqli_affected_rows()函数返回前一次 MySQL 操作所影响的行数。语法格式如下：

```
mysqli_affected_rows(connection)
```

其中 connection 为必需参数，表示当前的 MySQL 连接。如果返回结果为 0，表示没有受影响的记录；-1 表示查询返回错误。

下面通过实例来讲解它们的使用方法和区别。

【例 19.4】（实例文件：ch19\19.4.php）

```
<?php
$db=@mysqli_connect('localhost','root','753951','adatabase')
or die("无法连接到服务器");  //连接数据库
//执行查询数据操作
$sq = "select * from user";
$result = mysqli_query($db,$sq); //如果查询成功，$result 为资源类型，保存查询结果集
echo "查询结果有". mysqli_num_rows($result)."条记录" <br/>;//输出查询记录集的行数
// 执行更新数据操作
$sq = "update user set Name='mingming' where Name='张芳'";
$result = mysqli_query($db,$sq);
echo "更新了".mysqli_affected_rows($db). "条记录";  //输出更新记录集的行数
mysqli_close($db);
?>
```

程序执行后的结果如图 19-6 所示。

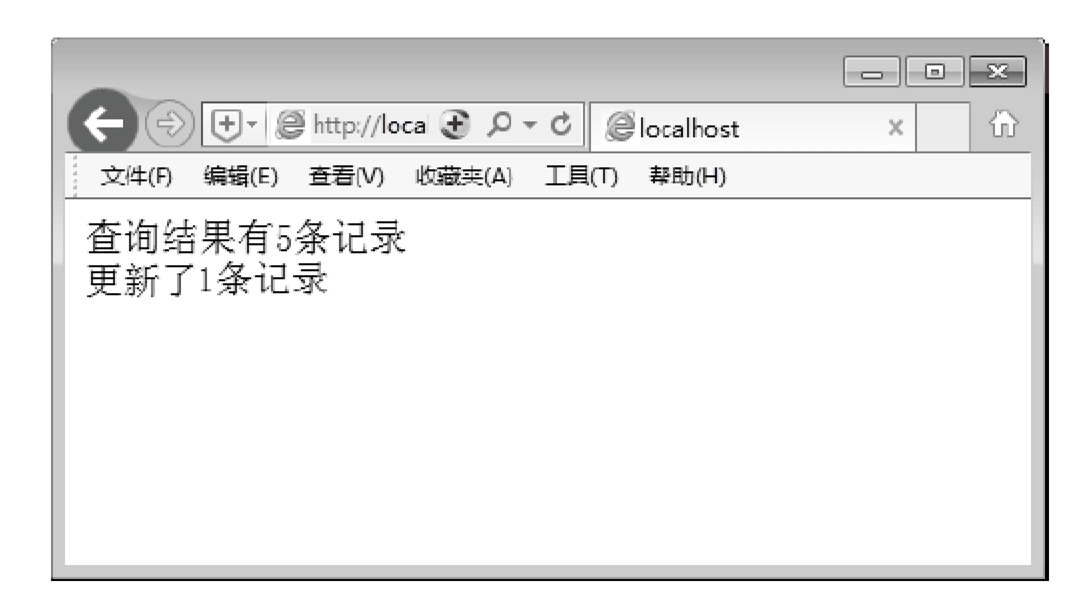


图 19-6　程序运行结果

19.3.6　获取结果集的一条记录作为枚举数组

执行 select 查询操作后，使用 mysqli_fetch_rows()函数可以从查询结果中取出数据，如果想逐行取出每条数据，可以结合循环语句循环输出。

mysqli_fetch_rows()函数的语法格式如下：

```
mysqli_ fetch_rows (result);
```

其中 result 指查询结果对象。

【例 19.5】（实例文件：ch19\19.5.php）

```
<?php
$db=@mysqli_connect('localhost','root','753951','adatabase')
or die("无法连接到服务器"); //连接数据库
mysqli_query("set names utf8"); //设置 MySQL 的字符集，以屏蔽乱码
//执行查询数据操作
$sq = "select * from user";
$result = mysqli_query($db,$sq); //如果查询成功，$result 为资源类型，保存查询结果集
<table width="370" border="1" cellspacing="0" cellpadding="0">
  <tr><th>编号</th><th>姓名</th><th>年龄</th><th>性别</th><th>个人信息/th></tr>
<?php
  while($row=mysqli_fetch_row($result)){  //逐行获取结果集中的记录，并显示在表格中
?>
  <tr>
    <td><?php echo $row[0] ?></td>          <!-- 显示第一列 -->
    <td><?php echo $row[1] ?></td>          <!-- 显示第二列 -->
    <td><?php echo $row[2] ?></td>          <!-- 显示第三列 -->
    <td><?php echo $row[3] ?></td>          <!-- 显示第四列 -->
    <td><?php echo $row[4] ?></td>          <!-- 显示第五列 -->
   </tr>
 <?php
  }
mysqli_close($db);
?>
```

程序执行后的结果如图 19-7 所示。

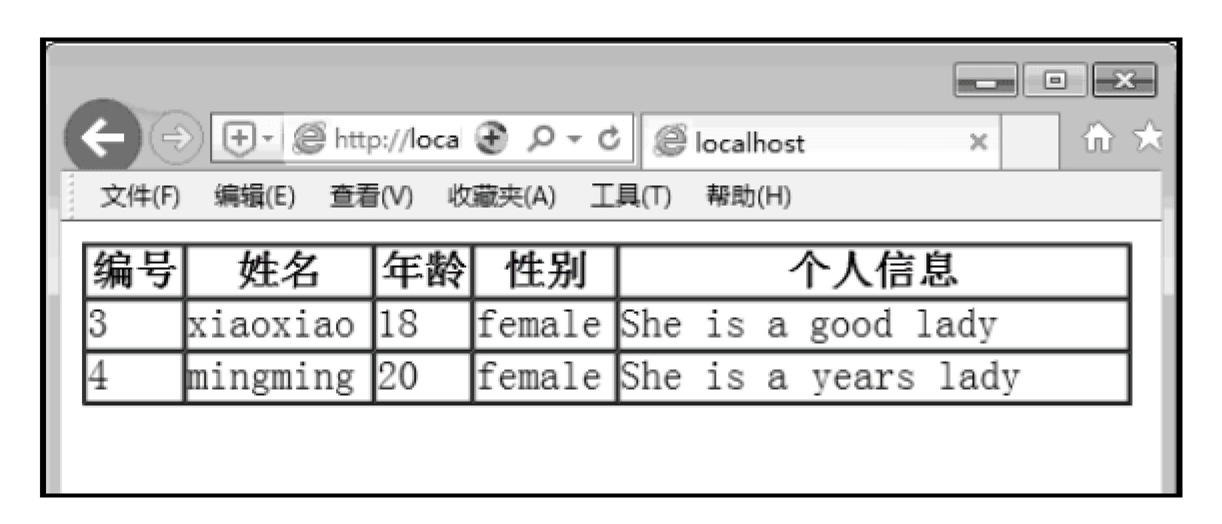


图 19-7　程序运行结果

19.3.7　获取结果集的记录作为关联数组

使用 mysqli_fetch_assoc()函数从数组结果集中获取信息，只要确定 SQL 请求返回的对象就可以了。语法格式如下：

```
mysqli_ fetch_assoc (result);
```

此函数与 mysqli_fetch_rows()函数的不同之处就是返回的每一条记录都是关联数组。注意，该函数返回的字段名是区分大小写的。

【例 19.6】（实例文件：ch19\19.6.php）

```
<?php
 while($row = mysqli_fetch_assoc($result)) {       // 逐行获取结果集中的记录
?>
 <tr>
   <td><?php echo $row["Id"] ?></td>          <!-- 获取当前行“Id”字段值 -->
   <td><?php echo $row["Name"] ?></td>     <!-- 获取当前行“Name”字段值 -->
   <td><?php echo $row["Age"] ?></td>      <!-- 获取当前行“Age”字段值 -->
   <td><?php echo $row["Gender"] ?></td>       <!-- 获取当前行“Gender”字段值
-->
   <td><?php echo $row["Info"] ?></td>     <!-- 获取当前行“Info”字段值 -->
 </tr>
 <?php
  }
?>
```

$row = mysqli_fetch_assoc($result);语句直接从$result 结果中取得一行，并且以关联数组的形式返回给$row。由于获得的是关联数组，所以在读取数组元素的时候要通过字段名称确定数组元素。

19.3.8　获取结果集中的记录作为对象

使用 mysqli_fetch_object()函数从结果中获取一行记录作为对象。语法格式如下：

```
mysqli_ fetch_ object (result);
```

【例 19.7】（实例文件：ch19\19.7.php）

```
<?php
```

```
  while($row = mysqli_fetch_object($result)) {      // 逐行获取结果集中的记录
?>
  <tr>
    <td><?php echo $row->Id ?></td>          <!-- 获取当前行“Id”字段值 -->
    <td><?php echo $row->Name ?></td>        <!-- 获取当前行“Name”字段值 -->
    <td><?php echo $row->Age ?></td>     <!-- 获取当前行“Age”字段值 -->
    <td><?php echo $row->Gender ?></td>      <!-- 获取当前行“Gender”字段值 -->
    <td><?php echo $row->Info ?></td>        <!-- 获取当前行“Info”字段值 -->
  </tr>
 <?php
  }
?>
```

该程序的整体运行结果和上一节案例相同。不同的是，这里的程序采用了对象和对象属性的表示方法。但是最后输出的数据结果是相同的。

19.3.9　使用 mysqli_fetch_array()函数获取结果集记录

mysqli_fetch_array ()函数的语法格式如下：

```
mysqli_fetch_ array (result[,resuilt_type])
```

参数 resuilt_type 是可选参数，表示一个常量，可以选择 MYSQL_ASSOC（关联数组），MYSQL_NUM（数字数组）和 MYSQL_BOTH（二者兼有），本参数的默认值为 MYSQL_BOTH。

【例 19.8】（实例文件：ch19\19.8.php）

```
<?php
  while($row = mysqli_fetch_array($result)) {       // 逐行获取结果集中的记录
?>
  <tr>
    <td><?php echo $row["Id"] ?></td>            // 使用字段名做索引显示字段值
    <td><?php echo $row["1"] ?></td>        // 使用数字做索引显示字段值
    <td><?php echo $row["Age"] ?></td>
    <td><?php echo $row["3"] ?></td>
    <td><?php echo $row["Info"] ?></td>
  </tr>
 <?php
  }
?>
```

19.3.10　使用 mysqli_free_result()函数释放资源

释放资源的函数为 mysqli_free_result()，函数的格式为：

```
mysqli_free_result(SQL 请求所返回的数据库对象)
```

A 在一切操作都基本完成以后，程序通过 mysqli_free_result($result);语句释放了 SQL 请求所返回的对象$result 所占用的资源。

19.4 实战演练 1——PHP 操作数据库

下面以通过 Web 向 user 数据库请求数据为例，介绍如何使用 PHP 函数处理 MySQL 数据库数据。具体步骤如下。

01 在网址主目录下创建 phpmysql 文件夹。

02 在 phpmysql 文件夹下建立文件 htmlform.html，输入如下代码。

```
<html>
<head>
   <title>Finding User</title>
</head>
<body>
  <h2>Finding users from mysql database.</h2>
  <form action="formhandler.php" method="post">
     Fill user name:
     <input name="username" type="text" size="20"/> <br />
    <input name="submit" type="submit" value="Find"/>
  </form>
</body>
</html>
```

03 在 phpmysql 文件夹下建立文件 formhandler.php，输入如下代码。

```
<html>
<head>
   <title>User found</title>
</head>
<body>
   <h2>User found from mysql database.</h2>
<?php
    $username = $_POST['username'];
     if(!$username){
         echo "Error: There is no data passed.";
          exit;
     }

     if(!get_magic_quotes_gpc()){
          $username = addslashes($username);
     }
          @ $db = mysqli_connect('localhost','root','753951','adatabase');

          if(mysqli_connect_errno()){
           echo "Error: Could not connect to mysql database.";
           exit;
          }

          $q = "SELECT * FROM user WHERE name = '".$username."'";

          $result = mysqli_query($db,$q);
          $rownum = mysqli_num_rows($result);
```

```
            for($i=0; $i<$rownum; $i++){
                $row = mysqli_fetch_assoc($result);
                echo "Id:".$row['id']."<br />";
                echo "Name:".$row['name']."<br />";
                echo "Age:".$row['age']."<br />";
                echo "Gender:".$row['gender']."<br />";
                echo "Info:".$row['info']."<br />";
            }
            mysqli_free_result($result);

            mysqli_close($db);

    ?>
    </body>
    </html>
```

04 运行 htmlform.html，结果如图 19-8 所示。

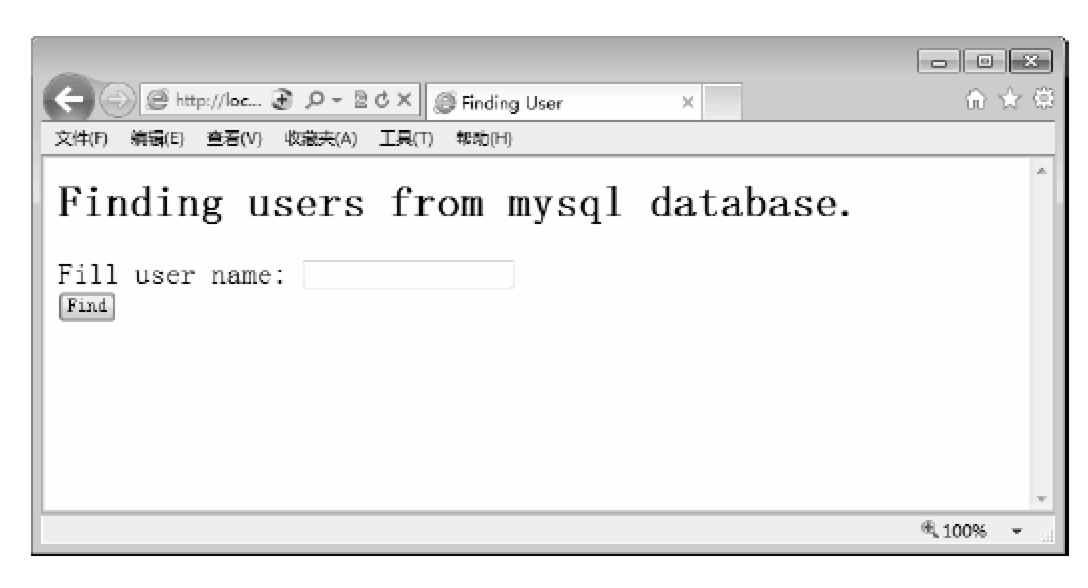


图 19-8　htmlform.html 页面

05 在输入框中输入用户名 lilili，单击 find 按钮，页面跳转至 formhandler.php，并且返回请求结果，如图 19-9 所示。



图 19-9　formhandler.php 页面

19.5　实战演练 2——使用 insert 语句动态添加用户信息

在前面的实例中，程序通过 form 查询了特定用户名的用户信息。下面将使用其他 SQL 语句实现 PHP 的数据请求。

下面通用使用 adatabase 的 user 数据库表格，添加新的用户信息。具体操作步骤如下。

01 在 phpmysql 文件夹下建立文件 insertform.html，并且输入代码如下。

```
<html>
<head>
```

```
    <title>Adding User</title>
  </head>
  <body>
    <h2>Adding users to mysql database.</h2>
    <form action="formhandler.php" method="post">
      Select gender:
        <select name="gender">
          <option value="male">man</option>
          <option value="female">woman</option>
        </select><br />
      Fill user name:
        <input name="username" type="text" size="20"/> <br />
      Fill user age:
        <input name="age" type="text" size="3"/> <br />
      Fill user info:
        <input name="info" type="text" size="60"/> <br />
        <input name="submit" type="submit" value="Add"/>
    </form>
  </body>
</html>
```

02 在 phpmysql 文件夹下建立文件 insertformhandler.php，并且输入代码如下。

```
<html>
<head>
   <title>User adding</title>
</head>
<body>
   <h2>adding new user.</h2>
<?php
    $username = $_POST['username'];
     $gender = $_POST['gender'];
     $age = $_POST['age'];
     $info = $_POST['info'];
     if(!$username and !$gender and !$age and !$info){
         echo "Error: There is no data passed.";
          exit;
     }
    if(!$username or !$gender or !$age or !$info){
         echo "Error: Some data did not be passed.";
          exit;
     }
     if(!get_magic_quotes_gpc()){
         $username = addslashes($username);
          $gender = addslashes($gender);
          $age = addslashes($age);
          $info = addslashes($info);
     }
          @ $db = mysqli_connect('localhost','root','753951', 'adatabase');
          if(mysqli_connect_errno()){
           echo "Error: Could not connect to mysql database.";
           exit;
          }
          $q = "INSERT INTO user( Name, Age, Gender, Info)
          VALUES ('$username',$age,'$gender', '$info')";
          if( !mysqli_query($db,$q)){
             echo "no new user has been added to database.";
          }else{
```

```
            echo "New user has been added to database.";
        };
        mysqli_close($db);
?>
</body>
</html>
```

03 运行 insertform.html，运行结果如图 19-10 所示。

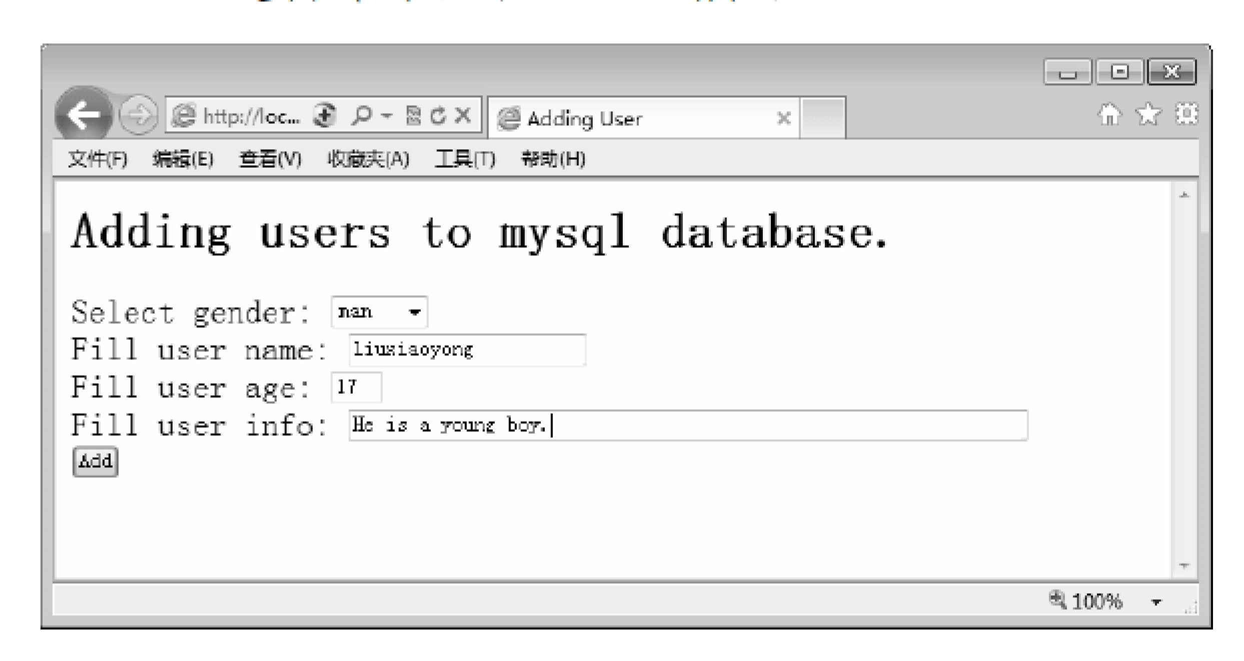


图 19-10 insertform.html 运行结果

04 单击 Add 按钮，页面跳转至 insertformhandler.php，并且返回信息结构，如图 19-11 所示。



图 19-11 insertformhandler.php 页面

【案例分析】：

这时数据库 user 表格中，就被添加了一个新的元素。

（1）insertform.html 文件中，建立了 user 表格中除 id 外每个字段的信息输入框。

（2）insertformhandler.php 文件中，建立 MySQL 连接，生成连接对象等操作都与前面的程序相同。只是改变了 SQL 请求语句的内容为$q = "INSERT INTO user(Name, Age, Gender, Info) VALUES ('$username',$age,'$gender', '$info')";插入语句。

（3）其中 Name、Gender、Info 字段为字符串型，所以'$username'、'$gender'、'$info' 3 个变量要以字符串形式加入。

19.6 实战演练 3——使用 select 语句查询数据信息

本案例讲述如何使用 select 语句查询数据信息。具体操作步骤如下。

01 在 phpmysql 文件夹下建立文件 selectform.html，并且输入代码如下。

```
<html>
<head>
   <title>Finding User</title>
</head>
<body>
  <h2>Finding users from mysql database.</h2>
  <form action="selectformhandler.php" method="post">
     Select gender:
      <select name="gender">
        <option value="male">man</option>
        <option value="female">woman</option>
      </select><br />
      <input name="submit" type="submit" value="Find"/>
  </form>
</body>
</html>
```

02 在 phpmysql 文件夹下建立文件 selectformhandler.php，并且输入代码如下。

```
<html>
<head>
   <title>User found</title>
</head>
<body>
   <h2>User found from mysql database.</h2>
<?php
    $gender = $_POST['gender'];
    if(!$gender){
        echo "Error: There is no data passed.";
        exit;
    }
    if(!get_magic_quotes_gpc()){
        $gender = addslashes($gender);
    }
        @ $db = mysqli_connect('localhost','root','753951');
        mysqli_select_db($db,'adatabase');
        if(mysqli_connect_errno()){
         echo "Error: Could not connect to mysql database.";
         exit;
        }
        $q = "SELECT * FROM user WHERE gender = '".$gender."'";
        $result = mysqli_query($db,$q);
        $rownum = mysqli_num_rows($result);
        for($i=0; $i<$rownum; $i++){
           $row = mysqli_fetch_assoc($result);
           echo "Id:".$row['id']."<br />";
           echo "Name:".$row['name']."<br />";
           echo "Age:".$row['age']."<br />";
           echo "Gender:".$row['gender']."<br />";
           echo "Info:".$row['info']."<br />";
        }
        mysqli_free_result($result);
        mysqli_close($db);
?>
</body>
</html>
```

03 运行 selectform.html，结果如图 19-12 所示。

04 单击 Find 按钮，页面跳转至 selectformhandler.php，并且返回信息，如图 19-13 所示。

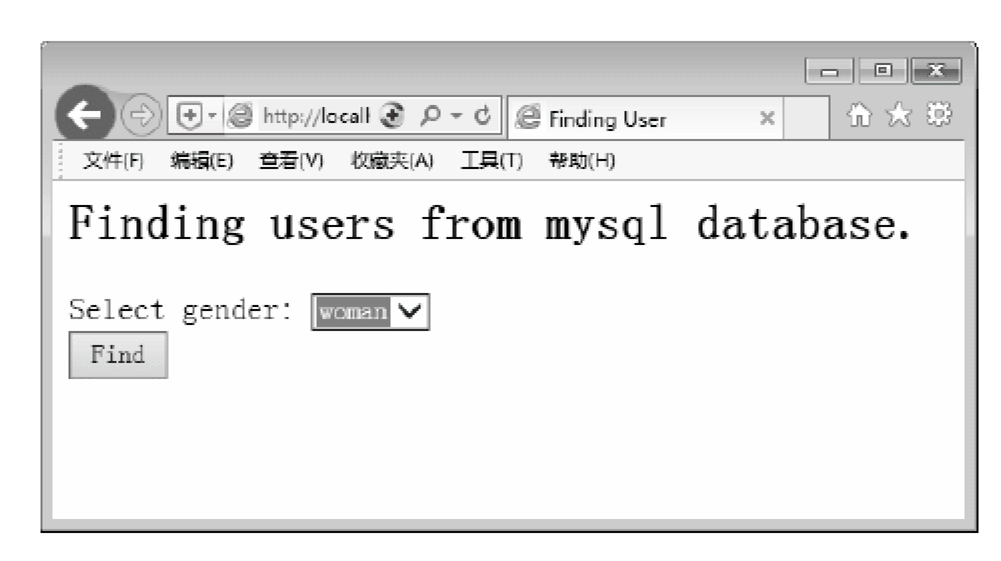


图 19-12　selectform.html 运行结果

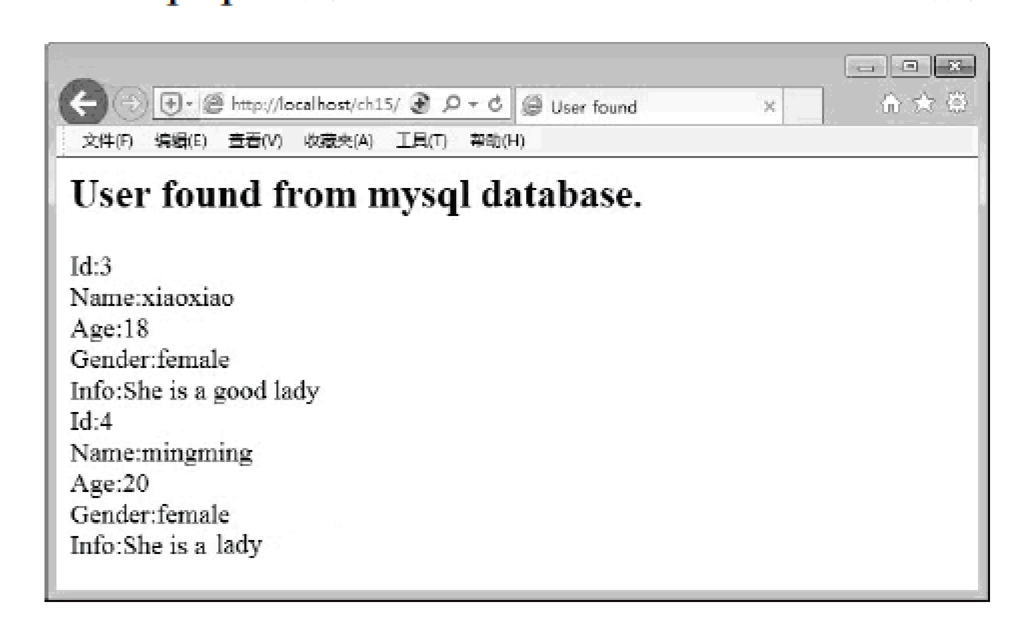


图 19-13　selectformhandler.php 页面

这样程序就给出了所有 gender 为 female 的用户信息。

19.7　高手私房菜

技巧 1：修改 php.ini 文件后仍然不能调用 MySQL 数据库怎么办？

有时候修改 php.ini 文件不能保证一定可以加载 MySQL 函数库。此时如果使用 phpinfo()函数不能显示 MySQL 的信息，说明配置失败了。重新按照 19.2 节的内容检查配置是否正确，如果正确，则把 PHP 安装目录下的 libmysql.dll 库文件直接复制到系统的 system32 目录下，然后重新启动 IIS 或 APACHE，最好再次使用 phpinfo()进行验证，即可看到 MySQL 信息，表示此时已经配置成功。

技巧 2：为什么尽量省略 MySQL 语句中的分号？

在 MySQL 语句中，每一行的命令都使用分号（；）作为结束，但是，当一行 MySQL 被插入在 PHP 代码中时，最好把后面的分号省略掉。这主要是因为 PHP 也是以分号作为一行的结束的，额外的分号有时会让 PHP 的语法分析器搞不明白，所以还是省略掉的好。在这种情况下，虽然省略了分号，但是 PHP 在执行 MySQL 命令时会自动加上。

另外还有不要加分号的情况。当用户想把字段竖着排列显示下来，而不是像通常那样横着排列时，可以用 G 来结束一行 SQL 语句，这时就用不上分号了，例如：

```
SELECT * FROM paper WHERE USER_ID =1G
```

19.8　经典习题

（1）在 Windows 系统下配置 PHP。

（2）编写连接 MySQL 数据库的代码。

（3）操作 test 数据库下的 fruits 数据表，包括查询、插入、更新和删除操作。

第 20 章　新闻发布系统数据库设计

MySQL 数据库的使用非常广泛，很多的网站和管理系统使用 MySQL 数据库存储数据。本章节主要讲述新闻发布系统的数据库设计过程。通过本章节的学习，读者可以在新闻发布系统的设计过程中学会如何使用 MySQL 数据库。

本章学习目标

- 了解新闻发布系统的概述
- 熟悉新闻发布系统的功能
- 掌握如何设计新闻发布系统的表
- 掌握如何设计新闻发布系统的索引
- 掌握如何设计新闻发布系统的视图
- 掌握如何设计新闻发布系统的触发器

20.1　系统概述

本章介绍的是一个小型新闻发布系统，管理员可以通过该系统发布新闻信息，管理新闻信息。一个典型的新闻发布系统网站至少应包含新闻信息管理、新闻信息显示和新闻信息查询 3 种功能。

新闻发布系统所要实现的功能具体包括：新闻信息添加、新闻信息修改、新闻信息删除、显示全部新闻信息、按类别显示新闻信息、按关键字查询新闻信息、按关键字进行站内查询。

本站为一个简单的新闻信息发布系统，该系统具有以下特点。

- 实用：系统实现了一个完整的信息查询过程。
- 简单易用：为使用户尽快掌握和使用整个系统，系统结构简单但功能齐全，简洁的页面设计使操作起来非常简便。
- 代码规范：作为一个实例，文中的代码规范简洁、清晰易懂。

本系统主要用于发布新闻信息、管理用户、管理权限、管理评论等功能。这些信息的录入、查询、修改和删除等操作都是该系统重点解决的问题。

本系统主要功能包括以下几点：

（1）具有用户注册及个人信息管理功能。

（2）管理员可以发布新闻、删除新闻。

（3）用户注册后可以对新闻进行评论、发表留言。

（4）管理员可以管理留言和对用户进行管理。

20.2 系统功能

新闻发布系统分为 5 个管理部分，即用户管理、管理员管理、权限管理、新闻管理和评论管理。本系统的功能模块如图 20-1 所示。

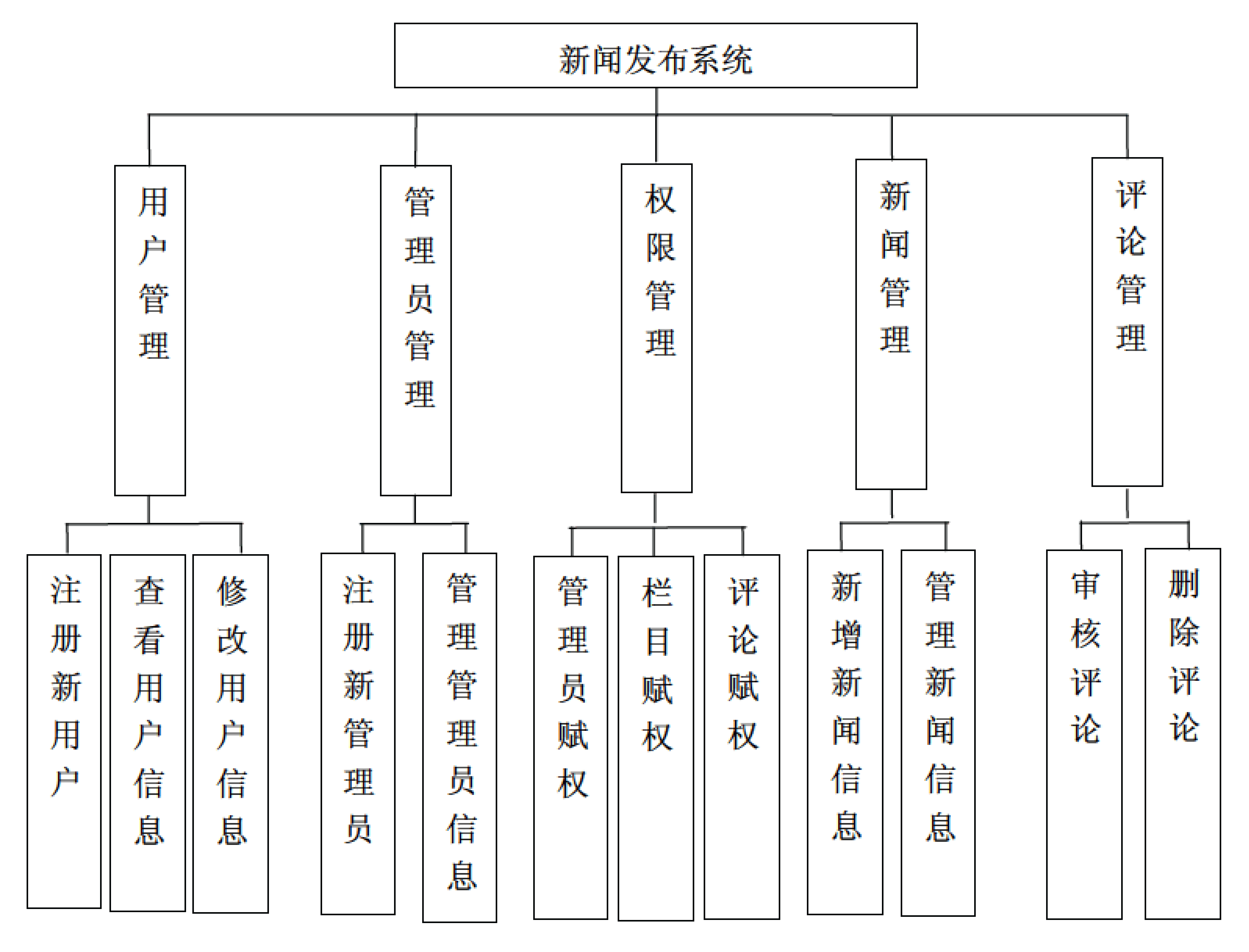


图 20-1 系统功能模块图

图 20-1 中模块的详细介绍如下。

（1）用户管理模块：实现新增用户，查看和修改用户信息功能。
（2）管理员管理模块：实现新增管理员，查看、修改和删除管理员信息功能。
（3）权限管理模块：实现对管理员、对管理的模块和管理的评论赋权功能。
（4）新闻管理模块：实现有相关权限的管理员对新闻的增加、查看、修改和删除功能。
（5）评论管理模块：实现有相关权限的管理员对评论的审核和删除功能。

通过本节的介绍，读者对这个新闻发布系统的主要功能有一定的了解，下一节会向读者介绍本系统所需要的数据库和表。

20.3 数据库设计和实现

数据库设计是开发管理系统的最重要的一个步骤。如果数据库设计得不够合理，将会为后续的开发工作带来很大的麻烦。本节为读者介绍新闻发布系统的数据库开发过程。

数据库设计时要确定设计哪些表、表中包含哪些字段、字段的数据类型和长度。通过本章节的学习，读者可以对 MySQL 数据库的知识有个全面的了解。

20.3.1 设计表

本系统所有的表都放在 webnews 数据库下。创建和选择 webnews 数据库的 SQL 代码如下：

```
CREATE DATABASE webnews;

USE webnews;
```

在这个数据库下总共存放 9 张表，分别是 user、admin、roles、news、category、comment、admin_Roles、news_Comment 和 users_Comment。

1. user 表

user 表中存储用户 ID、用户名、密码和用户 Email 地址，所以 user 表设计了 4 个字段。user 表每个字段的信息如表 20-1 所示。

表 20-1　user 表的内容

列名	数据类型	允许 NULL 值	说明
userID	INT	否	用户编号
userName	VARCHAR(20)	否	用户名称
userPassword	VARCHAR(20)	否	用户密码
userEmail	VARCHAR(20)	否	用户 Email

根据表 20-1 的内容创建 user 表。创建 user 表的 SQL 语句如下：

```
CREATE TABLE user(
        userID INT PRIMARY KEY UNIQUE NOT NULL,
        userName VARCHAR(20) NOT NULL,
        userPassword VARCHAR(20) NOT NULL,
        userEmail VARCHAR(20) NOT NULL
        );
```

2. admin 表

管理员信息表（admin）主要用来存放用户账号信息，如表 20-2 所示。

表 20-2　admin 表的内容

列名	数据类型	允许 NULL 值	说明
adminID	INT	否	管理员编号
adminName	VARCHAR(20)	否	管理员名称
adminPassword	VARCHAR(20)	否	管理员密码

根据表 20-2 的内容创建 admin 表。创建 admin 表的 SQL 语句如下：

```
CREATE TABLE admin(
        adminID INT PRIMARY KEY UNIQUE NOT NULL,
        adminName VARCHAR(20) NOT NULL,
        adminPassword VARCHAR(20) NOT NULL
        );
```

3. roles 表

权限信息表（roles）主要用来存放权限信息，如表 20-3 所示。

表 20-3　roles 权限信息表的内容

列名	数据类型	允许 NULL 值	说明
roleID	INT	否	权限编号
roleName	VARCHAR(20)	否	权限名称

根据表 20-3 的内容创建 roles 表。创建 roles 表的 SQL 语句如下：

```
CREATE TABLE roles(
        roleID INT PRIMARY KEY UNIQUE NOT NULL,
        roleName VARCHAR(20) NOT NULL
        );
```

4. news 表

新闻信息表（news）主要用来存放新闻信息，如图 20-4 所示。

表 20-4　news 新闻信息表的内容

列名	数据类型	允许 NULL 值	说明
newsID	INT	否	新闻编号
newsTitle	VARCHAR(50)	否	新闻标题
newsContent	TEXT	否	新闻内容
newsDate	TIMESTAMP	是	发布时间
newsDesc	VARCHAR(50)	否	新闻描述
newsImagePath	varchar(50)	是	新闻图片路径
newsRate	INT	否	新闻级别
newsIsCheck	BIT	否	新闻是否检验
newsIsTop	BIT	否	新闻是否置顶

根据表 20-4 的内容创建 news 表。创建 news 表的 SQL 语句如下：

```
CREATE TABLE news(
        newsID INT PRIMARY KEY UNIQUE NOT NULL,
        newsTitle VARCHAR(50) NOT NULL,
        newsContent TEXT NOT NULL,
        newsDate TIMESTAMP,
        newsDesc VARCHAR(50) NOT NULL,
        newsImagePath VARCHAR(50),
        newsRate INT NOT NULL,
        newsIsCheck BIT NOT NULL,
        newsIsTop BIT NOT NULL
        );
```

5. category 表

栏目信息表（categroy）主要用来存放新闻栏目信息，如图 20-5 所示。

表 20-5　category 栏目信息表的内容

列名	数据类型	允许 NULL 值	说明
categroyID	INT	否	栏目编号
categroyName	VARCHAR(50)	否	栏目名称
categroyDesc	VARCHAR(50)	否	栏目描述

根据表 20-5 的内容创建 categroy 表。创建 categroy 表的 SQL 语句如下：

```
CREATE TABLE categroy (
        categoryID INT PRIMARY KEY UNIQUE NOT NULL,
        categoryName VARCHAR(50) NOT NULL,
        categoryDesc VARCHAR(50) NOT NULL
        );
```

6. comment 表

评论信息表（comment）主要用来存放新闻评论信息，如图 20-6 所示。

表 20-6　comment 评论信息表的内容

列名	数据类型	允许 NULL 值	说明
commentID	INT	否	栏目编号
commentTitle	VARCHAR(50)	否	栏目标题
commentContent	VARCHAR(50)	否	栏目内容
commentDate	DATETIME	是	栏目时间

根据表 20-6 的内容创建 comment 表。创建 comment 表的 SQL 语句如下：

```
CREATE TABLE comment (
        commentID INT PRIMARY KEY UNIQUE NOT NULL,
        commentTitle VARCHAR(50) NOT NULL,
        commentContent TEXT NOT NULL,
        commentDate DATETIME
        );
```

7. admin_Roles 表

管理员_权限表（admin_Roles）主要用来存放管理员和权限的关系，如图 20-7 所示。

表 20-7　admin_Roles 管理员_权限表的内容

列名	数据类型	允许 NULL 值	说明
aRID	INT	否	管理员_权限编号
adminID	INT	否	管理员编号
roleID	INT	否	权限编号

根据表 20-7 的内容创建 admin_Roles 表。创建 admin_Roles 表的 SQL 语句如下：

```
CREATE TABLE admin_Roles (
        aRID INT PRIMARY KEY UNIQUE NOT NULL,
        adminID INT NOT NULL,
        roleID INT NOT NULL
        );
```

8. news_Comment 表

新闻_评论表（news_Comment）主要用来存放新闻和评论的关系，如图 20-8 所示。

表 20-8　新闻_评论表

列名	数据类型	允许 NULL 值	说明
nCommentID	INT	否	新闻_评论编号
newsID	INT	否	新闻编号
commentID	INT	否	评论编号

根据表 20-8 的内容创建 news_Comment 表。创建 news_Comment 表的 SQL 语句如下：

```
CREATE TABLE news_Comment (
        nCommentID INT PRIMARY KEY UNIQUE NOT NULL,
        newsID INT NOT NULL,
        commentID INT NOT NULL
        );
```

9. users_Comment 表

用户_评论表（users_Comment）主要用来存放用户和评论的关系，如图 20-9 所示。

表 20-9　新闻_评论表

列名	数据类型	允许 NULL 值	说明
uCID	INT	否	用户_评论编号
userID	INT	否	用户编号
commentID	INT	否	评论编号

根据表 20-8 的内容创建 users_Comment 表。创建 users_Comment 表的 SQL 语句如下：

```
CREATE TABLE news_Comment (
        uCID  INT PRIMARY KEY UNIQUE NOT NULL,
        userID  INT NOT NULL,
        commentID  INT NOT NULL
        );
```

创建完成后，可以使用 DESC 语句查看表的基本结构，也可以通过 SHOW CREATE TABLE 语句查看表的详细信息。

20.3.2　设计索引

索引是创建在表上的，是对数据库中一列或者多列的值进行排序的一种结构。索引可以提高查询的速度。新闻发布系统需要查询新闻的信息，这就需要在某些特定字段上建立索引，以便提高查询速度。

1. 在 news 表上建立索引

新闻发布系统中需要按照 newsTitle 字段、newsDate 字段和 newsRate 字段查询新闻信息。在本书的前面的章节中介绍了几种创建索引的方法。本小结将使用 CREATE INDEX 语句和 ALTER TABLE 语句创建索引。

下面使用 CREATE INDEX 语句在 newsTitle 字段上创建名为 index_new_title 的索引。SQL 语句如下：

```
CREATE INDEX index_new_title ON news(newsTitle);
```

然后，使用 CREATE INDEX 语句在 newsDate 字段上创建名为 index_new_date 的索引。SQL 语句如下：

```
CREATE INDEX index_new_date  ON news(newsDate);
```

最后，使用 ALTER TABLE 语句在 newsRate 字段上创建名为 index_new_rate 的索引。SQL 语句如下：

```
ALTER TABLE news  ADD INDEX index_new_rate (newsRate);
```

2. 在 categroy 表上建立索引

新闻发布系统中需要通过栏目名称查询该栏目下的新闻，因此需要在这个字段上创建索引。创建索引的语句如下：

```
CREATE INDEX index_categroy_name ON categroy (categroyName);
```

代码执行完成后，读者可以使用 SHOW CREATE TABLE 语句查看 categroy 表的详细信息。

3. 在 comment 表上建立索引

新闻发布系统需要通过 commentTitle 字段和 commentDate 字段查询评论内容。因此可以在这两个字段上创建索引。创建索引的语句如下：

```
CREATE INDEX index_comment _title  ON comment (commentTitle);
CREATE INDEX index_comment _date  ON comment (commentDate);
```

代码执行完成后，读者可以通过 SHOW CREATE TABLE 语句查看 comment 表的结构。

20.3.3　设计视图

视图是由数据库中一个表或者多个表导出的虚拟表。其作用是方便用户对数据的操作。在这个新闻发布系统中，也设计了一个视图改善查询操作。

在新闻发布系统中，如果直接查询 news_Comment 表，显示信息时会显示新闻编号和评论编号。

这种显示不直观，为了以后查询方面，可以建立一个视图 news_view。这个视图显示评论编号、新闻编号、新闻级别、新闻标题、新闻内容和新闻发布时间。创建视图 news_view 的 SQL 代码如下：

```
CREATE VIEW news_view
AS SELECT c.commentID,n.newsID,n.newsRate,n.newsTitle,n.newsContent,n.newsDate
FROM news_Comment c,news n
WHERE news_Comment.newsID=news.newsID;
```

SQL 语句中给每个表都取了别名，news_Comment 表的别名为 c；news 表的别名为 n，这个视图从这两个表中取出相应的字段。视图创建完成后，可以使用 SHOW CREATE VIEW 语句查看 news_view 视图的详细信息。

20.3.4　设计触发器

触发器是由 INSERT、UPDATE 和 DELETE 等事件来触发某种特定的操作。满足触发器的触发条件时，数据库系统就会执行触发器中定义的程序语句。这样做可以保证某些操作之间的一致性。为了使新闻发布系统的数据更新更加快速和合理，可以在数据库中设计几个触发器。

1. 设计 UPDATE 触发器

在设计表时，news 表和 news_Comment 表的 newsID 字段的值是一样的。如果 news 表中的 newsID 字段的值更新了，那么 news_Comment 表中的 newsID 字段的值也必须同时更新。这可以通过一个 UPDATE 触发器来实现。创建 UPDATE 触发器 update_newsID 的 SQL 代码如下：

```
DELIMITER &&
CREATE TRIGGER update_newsID AFTER UPDATE
        ON news FOR EACH ROW
        BEGIN
           UPDATE news_Comment SET newsID=NEW. newsID
        END
        &&
DELIMITER ;
```

其中 NEW. newsID 表示 news 表中更新的记录的 newsID 值。

2. 设计 DELETE 触发器

如果从 user 表中删除一个用户的信息，那么这个用户在 users_Comment 表中的信息也必须同时删除。这也可以通过触发器来实现。在 user 表上创建 delete_user 触发器，只要执行 DELETE 操作，那么就删除 users_Comment 表中相应的记录。创建 delete_user 触发器的 SQL 语句如下：

```
DELIMITER &&
CREATE TRIGGER delete_user AFTER DELETE
         ON user FOR EACH ROW
         BEGIN
            DELETE FROM users_Comment WHERE userID=OLD. userID
         END
         &&
DELIMITER ;
```

其中，OLD. userID 表示新删除的记录的 userID 值。

第 21 章　PHP+MySQL 开发论坛实战

PHP 与 MySQL 的结合是开发 Web 网站的黄金搭档。本章将以论坛网站的开发为例进行讲解。论坛网站的开发具有网站开发的代表性，通过本章的学习，读者可以掌握 PHP+MySQL 开发网站的常用知识和技巧。

本章学习目标

- 熟悉网站的需求分析
- 熟悉数据库分析
- 掌握论坛的实现过程

21.1　网站的需求分析

在开发网站之前，首先需要分析网站的需求，包括网站需求分析和网站的功能模板分析。

21.1.1　需求分析

需求分析是论坛网站开发的必要环节，下面分析论坛网站的需求如下。

（1）论坛的游客可以注册、登录网站和浏览主题。

（2）论坛的普通注册用户拥有浏览，发表主题、回复主题、修改自己的个人资料、查询主题、修改自己发布或回复的帖子等功能。

（3）版主对版块的管理功能，包括对帖子的主要操作有：查询主题、置顶、加精、移动、编辑和删除；对用户的操作有：禁止发言和删除 id；对版块的操作主要包括发布版块和广告。

（4）系统管理员对版块的操作有：建立、修改和删除版块；对用户的操作有：禁止发言和删除 id；对帖子的主要操作有：查询主题、置顶、加精、移动、编辑和删除；对论坛的操作有：开放或关闭会员注册功能。

21.1.2　网站功能模块分析

网站主要功能模块如下。

（1）会员注册模块：新会员注册，提供会员信息，检验会员信息的有效性，并将会员信息持久化。

（2）会员登录模块：提供用户凭证，验证用户信息，基于角色授权。

（3）会员管理模块：管理员由系统初始化分配，管理员可以对会员信息进行部分更改，主要包括用户角色调整、版主调整、删除会员等。

（4）论坛版块管理模块：管理员可以添加、删除、调整、置顶、隐藏论坛版块。

（5）帖子管理模块：管理员可以对所有帖子进行转移、置顶、删除等操作，版主可以对本版块帖子进行置顶、删除等操作。

（6）帖子发表模块：用户可以在其权限允许的版块内发表帖子。

（7）帖子回复模块：用户可以对其权限允许的主题发表回复。

（8）帖子浏览模块：用户可以浏览所有可见的帖子。

（9）帖子检索模块：注册用户可以提供标题关键字检索可见的主题帖，并可以查看自己发表或回复的帖子。

21.2　数据库分析

分析完网站的功能后，下面开始分析数据库的逻辑结构并建立数据表。

21.2.1　分析数据库

本论坛的数据库名称为 bbs_data，共有 5 个数据表，即 manage_user_info、user_info、father_module_info、son_module_info、note_info。各个数据表之间的逻辑关系如图 21-1 所示。

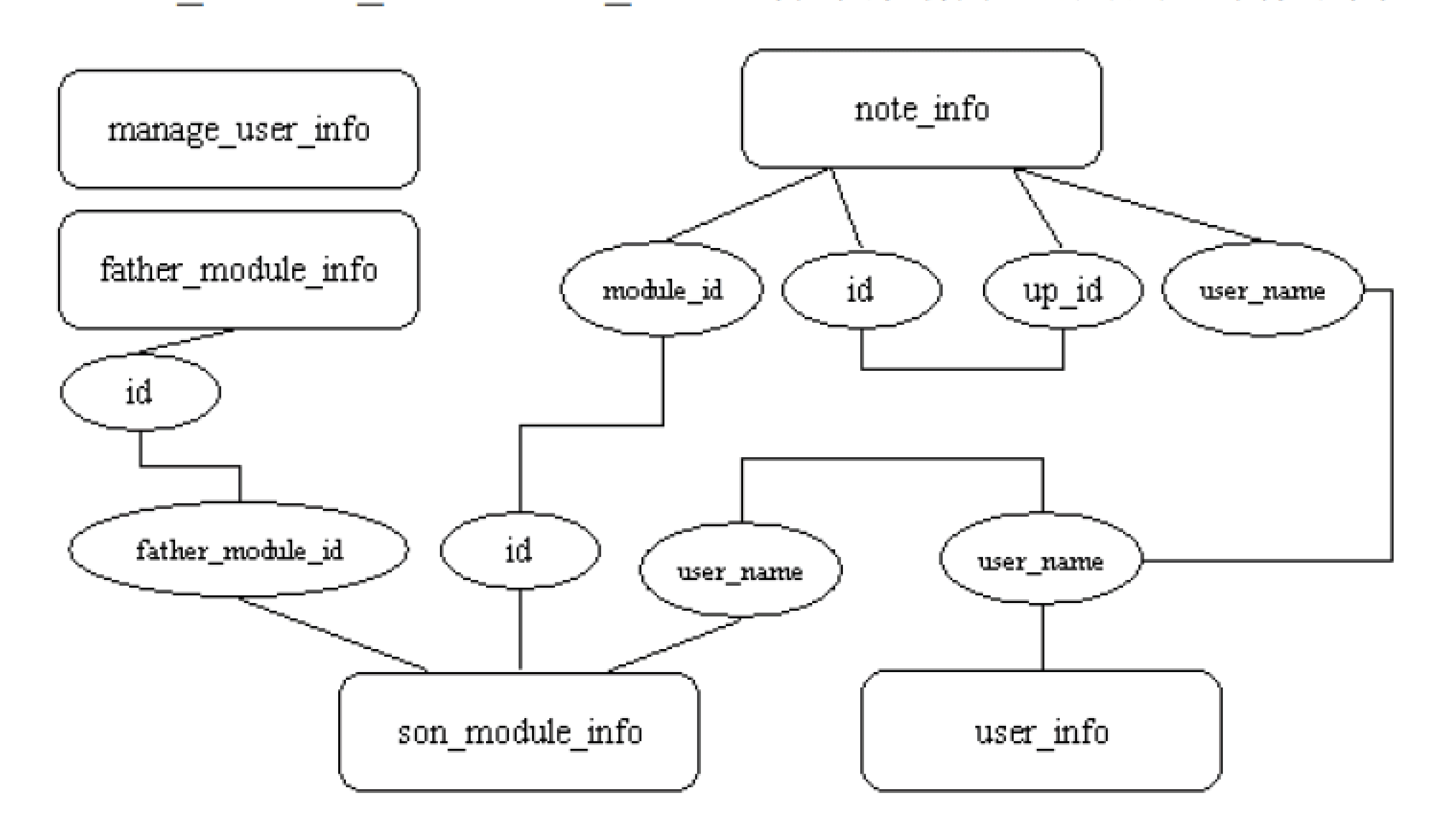


图 21-1　数据表的逻辑关系图

21.2.2　创建数据表

分析数据库的结构后，即可创建数据表，各个数据表如表 21-1~表 21-5 所示。

表 21-1　manage_user_info（管理用户信息数据表）

编号	字段名	类型	字段意义	备注
1	id	int		
2	user_name	char(16)	管理用户登录名	
3	user_pw	char(16)		

表 21-2　user_info（用户信息数据表）

编号	字段名	类型	字段意义	备注
1	id	int		
2	user_name	char(16)	管理用户登录名	
3	user_pw	char(16)		
4	time1	datetime	注册时间	
5	time2	datetime	最后登录时间	

表 21-3　father_module_info（父版块信息数据表）

编号	字段名	类型	字段意义	备注
1	id	int		1
2	module_name	char(66)	版块名称	2
3	show_order	int	显示序号	3

表 21-4　son_module_info（子版块信息数据表）

编号	字段名	类型	字段意义	备注
1	id	int		
2	father_module_id	int	隶属的父版块的 id	同 father_module_info 中的 id
3	module_name	char(66)	子版块名称	
4	module_cont	text	子版块简介	
5	user_name	char(16)	发帖用户名	同 user_info 中的 user_name

表 21-5　note_info（发帖信息数据表）

编号	字段名	类型	字段意义	备注
1	id	int		
2	module_id	int	隶属的子版块的 id	同 son_module_info 中的 id
3	up_id	int	回复帖子的 id	同本表中的 id
4	title	char(88)	帖子标题	
5	cont	text	帖子内容	
6	time	datetime	发帖时间	
7	user_name	char(16)	发帖用户名	同 user_info 中的 user_name
8	times	int	浏览次数	

21.3　论坛的代码实现

下面来分析论坛的代码是如何实现的。

21.3.1　数据库连接相关文件

主要的数据库连接相关文件如下。

文件 mysql.inc 位于配套素材包的 ch21\inc\下，主要用于自编连接数据库、服务器、SQL 语句执行函数库。主要代码如下。

```
<?php
class mysqlconn{
    //连接服务器、数据库以及执行 SQL 语句的类库
    public $database;
    public $server_username;
    public $server_userpassword;
    function mysqli()
     {  //构造函数初始化所要连接的数据库
      $this->server_username="root";
      $this->server_userpassword="";
      }//end mysqli()
```

```
    function link($database)
        {  //连接服务器和数据库
         //设置所有连接的数据库
          if ($database==""){
             $this->database="bbs_data";
             }else{
             $this->database=$database;
             }
          //连接服务器和数据库

    if(@$id=mysqli_connect('localhost',$this->server_username,$this->server_us
erpassword)){
            if(!mysqli_select_db($id,$this->database)){
              echo "数据库连接错误!!! ";
                exit;
                }
              }else{
                echo "服务器正在维护中，请稍后重试!!! ";
                exit;
               }
    }//end link($database)
    function excu($query)
      {     //执行 SQL 语句
            if($result=mysqli_query($query)){
                return $result;
            }else{
                echo "sql 语句执行错误!!! 请重试!!!";
                    exit;
            }
      } //end  exec($query)
   } //end class mysql
   ?>
```

文件 myfunction.inc 位于配套素材包的 ch21\inc\下，主要用于自编函数库。主要代码如下。

```
   <?php
   class myfunction{
   ////////////////字符转换：向数据库中插入或更新时用////////////////////////
        function str_to($str)
        {
          $str=str_replace(" "," ",$str);      //把空格替换 html 的字符串空格
          $str=str_replace("<","&lt;",$str);        //把 html 的输出标志正常输出
          $str=str_replace(">","&gt;",$str);        //把 html 的输出标志正常输出
          $str=nl2br($str);                      //把回车替换成 html 中的 br
          return $str;
            }
```

```
    /////////////////////由子版块的 id 返回该子版块的主题数//////////////////////
     function son_module_idtonote_num($son_module_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where module_id='".$son_module_id."'
and up_id='0'";
            $rst=$aa->excu($query);
            return mysqli_num_rows($rst);
     }
    ////////////////////////////////////////////////////////////////////////////
/
    /////////////////////由子版块的 id 返回该子版块的帖子数//////////////////////
        function son_module_idtonote_num2($son_module_id){
        //由子版块的 id 返回该子版块的主题数
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where module_id='".$son_module_id."'
and up_id='0'";
            $rst=$aa->excu($query);
            $num=mysqli_num_rows($rst);
            while ($note=mysqli_fetch_array($rst,MYSQL_ASSOC)){
                $query="select * from note_info where up_id='".$note['id']."' and
module_id='0'";
                $rst=$aa->excu($query);
                $num+=mysqli_num_rows($rst);
                }
            return $num;
     }
    /////////////////////由子版块的 id 输出该子版块的最新帖子////////////////////
        function son_module_idtolast_note($son_module_id){
        //由子版块的 id 输出该子版块的最新帖子
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where module_id='".$son_module_id."'
order by time desc limit 0,1";
            $rst=$aa->excu($query);
            $note=mysqli_fetch_array($rst,MYSQL_ASSOC);
                $query2="select * from note_info where id='".$note['up_id']."'";
                $rst2=$aa->excu($query);
                $note2=mysqli_fetch_array($rst2,MYSQL_ASSOC);
                echo $note2['title'];
                echo "<br />";
                echo $note['time']."  ".$note['user_name'];
     }
    /////////////////////由子版块的 id 输出该子版块的版主/////////////////////
```

```
        function son_module_idtouser_name($son_module_id){
         //由子版块的 id 输出该子版块的版主
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from son_module_info where id='".$son_module_id."'";
            $rst=$aa->excu($query);
            $module=mysqli_fetch_array($rst,MYSQL_ASSOC);
            if ($module['user_name']==""){
                return "版主暂缺";
                }else{
                    return $module['user_name'];
                }
    }
    ///////////////输出所有版块的下拉列表（子版块有参数）///////////////////
        function son_module_list($son_module_id){
         //输出所有版块的下拉列表（子版块有参数）
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from father_module_info order by id";
            $rst=$aa->excu($query);
            echo "<select name=module_id>";
            while($father_module=mysqli_fetch_array($rst,MYSQL_ASSOC)){
                echo "<option value=>".$father_module['module_name']."</option>";
                $query="select * from son_module_info where father_module_id='".
$father_module['id']."' order by id ";
                $rst2=$aa->excu($query);
                while($son_module=mysqli_fetch_array($rst2,MYSQL_ASSOC)){
                    echo"<option
   value=".$son_module['id'].">  ".$son_module['module_name']."</option>";
                    }
                }
            echo "</select>";
    }
    ////////////////////输出父版块的下拉列表//////////////////////////////////
        function father_module_list($father_module_id){
         //输出父版块的下拉列表
            $aa=new mysqlconn;
            $aa->link("");
            echo "<select name=father_module_id>";
            if ($father_module_id==""){
                echo "<option selected>请选择...</option>";
                }else{
                    $query="select     *     from     father_module_info     where
id='$father_module_id'";
                    $rst=$aa->excu($query);
```

```
                    $father_module=mysqli_fetch_array($rst,MYSQL_ASSOC);
                    echo"<option
    value=".$father_module['id'].">".$father_module['module_name']."</option>";
                    }
            $query="select * from father_module_info order by show_order";
            $rst=$aa->excu($query);
            while($father_module=mysqli_fetch_array($rst,MYSQL_ASSOC)){
                echo                                                    "<option
value=".$father_module['id'].">".$father_module['module_name']."</option>";
                }
            echo "</select>";
    }
    ////////////////////由帖子的 id 返回该帖子被浏览的次数//////////////////////
    function note_idtotimes($note_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where id='".$note_id."'";
            $rst=$aa->excu($query);
            $note=mysqli_fetch_array($rst,MYSQL_ASSOC);
            return $note['times'];
    }
    ////////////////////由帖子的 id 返回该帖子的标题//////////////////////
    function note_idtotitle($note_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where id='$note_id'";
            $rst=$aa->excu($query);
            $note=mysqli_fetch_array($rst,MYSQL_ASSOC);
            return $note['title'];
    }
    ////////////////////由帖子的 id 返回帖子的回复数/////////////////////
    function note_idtonote_num($note_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where up_id='".$note_id."'";
            $rst=$aa->excu($query);
            $num=mysqli_num_rows ($rst);
            return $num+1;
    }
    //////////////////由帖子的 id 输出帖子的最后回复时间///////////////////
    function note_idtolast_time($note_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where up_id='$note_id' order by time
desc limit 0,1";
```

```
            $rst=$aa->excu($query);
            $note=mysqli_fetch_array($rst,MYSQL_ASSOC);
            echo $note['time'];
    }
    ////////////////////由帖子的 id 输出帖子的最后回复人/////////////////////
    function note_idtolast_user_name($note_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info where up_id='$note_id' order by time
desc limit 0,1";
            $rst=$aa->excu($query);
            $note=mysqli_fetch_array($rst,MYSQL_ASSOC);
            echo $note['user_name'];
    }
    ////////////////////由子版块的 id 返回其父版块的名称/////////////////////
    function son_module_idtofather_name($son_module_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from son_module_info where id='$son_module_id'";
            $rst=$aa->excu($query);
            $module=mysqli_fetch_array($rst,MYSQL_ASSOC);
            $query2="select        *        from        father_module_info        where
id='$module[father_module_id]'";
            $rst2=$aa->excu($query2);
            $module2=mysqli_fetch_array($rst2,MYSQL_ASSOC);
            return $module2['module_name'];
    }
    ////////////////////由子版块的 id 返回本版块的名称////////////////////
    function son_module_idtomodule_name($son_module_id){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from son_module_info where id='".$son_module_id."'";
            $rst=$aa->excu($query);
            $module=mysqli_fetch_array($rst,MYSQL_ASSOC);
            return $module['module_name'];
    }
    ////////////////////所有帖子的总数////////////////////
    function note_total_num(){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from note_info";
            $rst=$aa->excu($query);
            return mysqli_num_rows ($rst);
    }
    ////////////////////所有会员的总数////////////////////
```

```
    function user_total_num(){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from user_info";
            $rst=$aa->excu($query);
            return mysqli_num_rows ($rst);
    }
   /////////////////////所有会员的总数//////////////////////
    function last_username(){
            $aa=new mysqlconn;
            $aa->link("");
            $query="select * from user_info order by id desc limit 0,1";
            $rst=$aa->excu($query);
            $user=mysqli_fetch_array($rst,MYSQL_ASSOC);
            return $user['user_name'];
    }
   /////////////分页函数///////////////////
         function page($query,$page_id,$add,$num_per_page){
        //  include "mysql.inc";
        ////////使用方法为:
        ///////  $myf=new myfunction;
        ///////  $query="";
     ///////  $myf->page($query,$page_id,$add,$num_per_page);
        ///////  $bb=$aa->excu($query);
          $bb=new mysqlconn;
          global $query;      //声明全局变量
          $bb->link("");
          $page_id=@$_GET['page_id'];    //接受page_id
          if ($page_id==""){
              $page_id=1;
              }
          $rst=$bb->excu($query);
          $num=mysqli_num_rows ($rst);
          if ($num==0){
              echo "没有查到相关记录或没有相关回复！<br />";
              }
          $page_num=ceil($num/$num_per_page);
          for ($i=1;$i<=$page_num;$i++){
              echo " [<a href=?".$add."page_id=".$i.">".$i."</a>]";
              }
          $page_up=$page_id-1;
          $page_down=$page_id+1;
          if ($page_id==1){
              echo "<a href=?".$add."page_id=".$page_down.">下一页</a>  
第".$page_id."页,共".$page_num."页";
```

```
            }
        else if ($page_id>=$page_num-1){
            echo "<a href=?".$add."page_id=".$page_up.">上一页</a>  
第".$page_id."页,共".$page_num."页";
            }
        else{
            echo   "<a   href=?".$add."page_id=".$page_up.">   上   一   页
</a>  <a href=?".$add."page_id=".$page_down.">下一页</a>  第
".$page_id."页,共".$page_num."页";
            }
        $page_jump=$num_per_page*($page_id-1);
        $query=$query." limit $page_jump,$num_per_page";
        }
    }//end myfunction
?>
```

21.3.2　论坛主页面

论坛主页面的相关文件如下。

文件 head.php 位于配套素材包的 ch21\inc\下，为论坛的头文件，代码如下。

```
<?php
@session_start();
?>
<style type="text/css">
<!--
@font-face {
  font-family: 'Hanyihei';
  src: url("inc/hanyihei.ttf") format("truetype");
  font-style: normal; }
@font-face {
  font-family: 'Minijanxixingkai';
  src: url("inc/minijanxixingkai.ttf") format("truetype");
  font-style: normal; }
.STYLE1 {
 font-family: 'Hanyihei';
 font-size: 36px;
       color:#024f6c;
}
.STYLE2 {
      font-family: 'Hanyihei';
}
-->
</style>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="1">
  <tr>
```

```
    <td height="60" bgcolor="f0b604"><span class="STYLE1">  迅捷 BBS
系统</span></td>
   </tr>
   <tr>
    <td height="2"></td>
   </tr>
  </table>
```

文件 foot.php 位于配套素材包的 ch21\inc\下，为论坛的版权文件，具体代码如下。

```
  <table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
   <tr>
    <td height="10"></td>
   </tr>
   <tr>
    <td height="10" bgcolor="#5F8AC5"></td>
   </tr>
   <tr>
    <td height="40" align="center" valign="middle" bgcolor="#f0b604">建议使用
浏览器 IE 6.0以上 分辨率1024*768以上<br />
      版权所有：<a href="http://www.quickbbs.net" target="_blank">迅捷 BBS</a>
</td>
   </tr>
  </table>
```

文件 total_info.php 位于配套素材包的 ch21\inc\下，为论坛的总信息文件。具体代码如下。

```
  <?php
  //@session_start();
  //用户登录并注册 SESSION
  if(isset($_POST['tijiao'])){
     $tijiao=$_POST['tijiao'];
  }
  if (@$tijiao=="提交"){
        $user_name=@$_POST['user_name'];
        $user_pw=@$_POST['user_pw'];
        $check_query="select * from user_info where user_name='".$user_name."'";
        $check_rst=$aa->excu($check_query);
        $user=mysqli_fetch_array($check_rst);
              if ($user_pw==$user['user_pw']){
              $_SESSION['user_name']=$user['user_name'];
              $today=date("Y-m-d H:i:s");
              $query="update user_info set time2='".$today."'
  where user_name='".$_SESSION['user_name']."'";
              $aa->excu($query);
              }
  }
```

```
    if (@$tijiao=="安全退出"){
    $_SESSION['user_name']="";
    }
    ?>
    <table width="98%" border="0" align="center" cellpadding="0" cellspacing="1">
      <tr><form id="form1" name="form1" method="post" action="#">
        <td      width="80%"      height="25"      align="left"      valign="middle"
bgcolor="5F8AC5"> 
     <?php
     if (@$_SESSION['user_name']!=""){
     echo "<font color=ffffff>欢迎您:".$_SESSION['user_name']."</font>";
     echo "         
    <input type='submit' name='tijiao' value='安全退出'>";
     }else{
     ?>
           用户名:
           <input type="text" name="user_name" size="8" />
           密码:<input type="text" name="user_pw" size="8" />
           <input type="submit" name="tijiao" value="提交" />
         <a  href="register.php"><font  color="#FFFFFF"> 我 要 注 册
</font></a>
        <?php
     }
     ?>
        </td>
     </form>
        <td width="20%" align="right" valign="middle" bgcolor="5F8AC5">
     <?php
      $today=date("Y-m-d H:i:s");
      echo $today;
     ?>
     </td>
      </tr>
      <tr>
        <td height="25" colspan="2" align="right" valign="middle">帖子总数：<?php
    echo $bb->note_total_num();?>  会员总数：<?php
    echo $bb->user_total_num();?>  欢迎新会员：
    <?php echo $bb->last_username();?>  </td>
      </tr>
      <tr>
        <td height="13" colspan="2" align="right" valign="middle"> </td>
      </tr>
    </table>
```

文件 index.php 位于配套素材包的 ch21\下，是用户访问的主页。具体代码如下。

```
    <html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>===迅捷 BBS 系统===</title>
    <link href="inc/style.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
    <?php
    @session_start();
    include "inc/mysql.inc";
    include "inc/myfunction.inc";
    include "inc/head.php";
    $aa=new mysqlconn;
    $bb=new myfunction;
    $aa->link("");
    include "inc/total_info.php";
    ?>
    <table class='indextemp' width="98%" border="0" align="center" cellpadding="0"
    cellspacing="1" bgcolor="#FFFFFF">
      <tr>
        <td    width="50%"    height="25"    align="center"    valign="middle"
bgcolor="5F8AC5">
    <span class="STYLE2">讨论区</span></td>
        <td  width="10%"  align="center"  valign="middle"  bgcolor="5F8AC5"><span
class="STYLE2">
    主 题</span></td>
        <td  width="10%"  align="center"  valign="middle"  bgcolor="5F8AC5"><span
class="STYLE2">
    帖 子</span></td>
        <td  width="20%"  align="center"  valign="middle"  bgcolor="5F8AC5"><span
class="STYLE2">
    最新帖子</span></td>
        <td  width="10%"  align="center"  valign="middle"  bgcolor="5F8AC5"><span
class="STYLE2">
    版 主</span></td>
      </tr>
      <tr>
        <td colspan="5">
     <?php
     $query="select * from father_module_info order by id";
     $result=$aa->excu($query);
     while($father_module=mysqli_fetch_array($result)){
     ?>
     <table width="100%" border="0" cellspacing="0" cellpadding="0">
          <tr>
```

```
        <td height="25" colspan="6" bgcolor="98B2CC">    
   <img src="pic_sys/li-1.gif" width="16" height="15">    
   <?php echo $father_module['module_name']?></td>
        </tr>
        <?php
     $query2="select * from son_module_info where father_module_id='".
$father_module['id']."'order by id";
     $result2=$aa->excu($query2);
     while($son_module=mysqli_fetch_array($result2)){
     ?>
     <tr>
         <td  width="5%"  height="40"  align="center"  valign="middle"><img
src="pic_sys/li-2.gif"
   width="32" height="32"></td>
         <td width="45%" align="left" valign="middle">
       <?php
       echo "<b><a href=module_list.php?module_id=".$son_module['id'].">
   <font color=0000ff>".$son_module["module_name"]."</font></a></b><br />";
       echo $son_module["module_cont"];
       ?>      </td>
         <td width="10%" align="center" valign="middle"><?php
   echo $bb->son_module_idtonote_num($son_module["id"]);?></td>
         <td width="10%" align="center" valign="middle"><?php
    echo $bb->son_module_idtonote_num2($son_module["id"]);?></td>
         <td width="20%" align="left" valign="middle"><?php
    echo $bb->son_module_idtolast_note($son_module["id"]);?></td>
         <td width="10%" align="center" valign="middle"><?php
    echo $bb->son_module_idtouser_name($son_module["id"]);?></td>
        </tr>
     <?php }?>
      </table>
    <?php } ?>
    </td>
     </tr>
   </table>
   <?php
   include "inc/foot.php";
   ?>
   </body>
   </html>
```

主页面运行后效果如图 21-2 所示。



图 21-2　论坛主页面

21.3.3　新用户注册页面

文件 register.php 位于配套素材包的 ch21\下，是新用户注册页面。具体代码如下。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>===迅捷 BBS 系统===</title>
<link href="inc/style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<?php
include "inc/mysql.inc";
include "inc/myfunction.inc";
include "inc/head.php";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
include "inc/total_info.php";
?>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="73%" height="30"><a href="./">迅捷 B B S 系统</a>>>新用户注册</td>
    <td width="27%" align="right" valign="middle"><a href="new_note.php"></a></td>
  </tr>
</table>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="1"
bgcolor="#FFFFFF">
  <tr>
    <td height="25" align="center" valign="middle" bgcolor="5F8AC5">发布新帖</td>
```

```
     </tr>
     <tr>
       <td height="25" align="center" valign="middle">
    <?php
    //接收提交表单内容检验数据库中是否已经存在此用户名，不存在写入数据库
    $tijiao=@$_POST['tijiao'];
    if ($tijiao=="提交"){
        $user_name=@$_POST['user_name'];
        $query="select * from user_info where user_name='$user_name'";
        $rst=$aa->excu($query);
        if (mysqli_num_rows($rst)!=0){
            echo "===您注册的用户名已经存在，请选择其他的用户名重新注册！===";
        }else{
            $user_pw1=$_POST['user_pw1'];
            $user_pw2=$_POST['user_pw2'];
            if ($user_pw1!=$user_pw2){
                echo "===您两次输入的密码不匹配，请重新输入！===";
            }else{
                $today=date("Y-m-d H:i:s");
                $query="insert into user_info
    (user_name,user_pw,time1) values('$user_name','$user_pw1','$today')";
                if ($aa->excu($query)){
                    echo "===恭喜您，注册成功！请<a href=../>
    返回主页</a>登录===";
                    $register_tag=1;
                }
            }
        }
    }
    //显示注册表单
    if (@$register_tag!=1){
    ?>
    <form name="form1" method="post" action="#">
    <table width="500" border="0" cellpadding="0" cellspacing="2">
        <tr>
          <td    width="121"    height="26"    align="right"    valign="middle"
bgcolor="#CCCCCC">
    用户名:</td>
          <td    width="372"    height="26"    align="left"    valign="middle"
bgcolor="#CCCCCC"><input
    type="text" name="user_name"></td>
        </tr>
        <tr>
          <td height="26" align="right" valign="middle" bgcolor="#CCCCCC"> 密
码:</td>
          <td height="26" align="left" valign="middle" bgcolor="#CCCCCC">
    <input type="text" name="user_pw1"></td>
        </tr>
        <tr>
```

```
          <td height="26" align="right" valign="middle" bgcolor="#CCCCCC">重复密
码:</td>
          <td height="26" align="left" valign="middle" bgcolor="#CCCCCC">
   <input type="text" name="user_pw2"></td>
        </tr>
        <tr>
          <td    height="26"    colspan="2"    align="center"    valign="middle"
bgcolor="#CCCCCC">
   <input type="submit" name="tijiao" value="提 交">    
   <input type="reset"
   name="Submit2" value="重 置"></td>
          </tr>
       </table>
    </form>
    <?php
    }
    ?>
    </td>
     </tr>
     <tr>
       <td height="1" bgcolor="#CCCCCC"></td>
     </tr>
   </table>
   <?php
   include "inc/foot.php";
   ?>
   </body>
   </html>
```

注册页面的运行效果如图 21-3 所示。输入用户名和密码后，单击【提交】按钮即可注册新用户。



图 21-3　新用户注册页面

注册完成后，即可在主页面输入用户名和密码，单击【提交】按钮，登录论坛系统。登录后效果如图 21-4 所示。单击【安全退出】按钮，即可退出登录操作。



图 21-4　用户成功登录页面

21.3.4　论坛帖子的相关页面

下面介绍论坛帖子的相关页面。

文件 new_note.php 位于配套素材包的 ch21\下，是用于发布新帖的页面。具体代码如下。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>===迅捷 BBS 系统===</title>
<link href="inc/style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<?php
//@session_start();
include "inc/mysql.inc";
include "inc/myfunction.inc";
include "inc/head.php";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
include "inc/total_info.php";
?>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="73%" height="30"><a href="./">迅捷 B B S 系统</a>>>发新帖子</td>
    <td width="27%" align="right" valign="middle"><a href="new_note.php">
```

```
    <img src="pic_sys/post.gif" width="82" height="20"></a></td>
      </tr>
    </table>
    <table width="98%" border="0" align="center" cellpadding="0" cellspacing="1"
bgcolor="#FFFFFF">
      <tr>
        <td height="25" align="center" valign="middle" bgcolor="5F8AC5">发 布 新 帖
</td>
      </tr>
      <tr>
        <td height="25" align="center" valign="middle">
     <?php
         if (@$_SESSION['user_name']==""){
     echo "===请先登录！==";
     }else{
     //接收提交表单内容写入数据库
     $tijiao=@$_POST['tijiao'];
     if ($tijiao=="提交"){
         $module_id=@$_POST['module_id'];
         $title=@$_POST['title'];
         $cont=@$_POST['cont'];
         $cont=$bb->str_to($cont);
         $today=date("Y-m-d H:i:s");
         if ($module_id!="" and $title!="" and $cont!=""){
             $query="insert into note_info
    (module_id,title,cont,time,user_name)
    values('$module_id','$title','$cont','$today','".$_SESSION['user_name']."')";
             if ($aa->excu($query)){
             echo "===新帖发布成功，请继续！===";
             }
         }else{
             echo "===请选择子模块，而且标题和内容均不能为空！===";
         }
     }
     ?>
     <form name="form1" method="post" action="new_note.php">
     <table width="500" border="0" cellpadding="0" cellspacing="2">
         <tr>
           <td     width="122"    height="26"    align="right"    valign="middle"
bgcolor="#CCCCCC">隶属版块:</td>
           <td     width="372"     height="26"     align="left"     valign="middle"
bgcolor="#CCCCCC">
         <?php
         $bb->son_module_list("");
         ?>
```

```
        </td>
        </tr>
        <tr>
          <td  height="26"  align="right"  valign="middle"  bgcolor="#CCCCCC"> 标
题:</td>
          <td height="26" align="left" valign="middle" bgcolor="#CCCCCC">
    <input type="text" name="title"></td>
        </tr>
        <tr>
          <td  height="26"  align="right"  valign="middle"  bgcolor="#CCCCCC"> 内
容:</td>
          <td height="26" align="left" valign="middle" bgcolor="#CCCCCC">
    <textarea name="cont" cols="50" rows="8"></textarea></td>
        </tr>
        <tr>
          <td height="26"  align="right"  valign="middle"  bgcolor="#CCCCCC">发帖
人:</td>
           <td height="26" align="left" valign="middle" bgcolor="#CCCCCC"> 
    <?php echo $_SESSION['user_name'];?></td>
        </tr>
        <tr>
          <td  height="26"  align="right"  valign="middle"  bgcolor="#CCCCCC"> 时
间:</td>
          <td height="26" align="left" valign="middle" bgcolor="#CCCCCC">系统将
自动记录! </td>
        </tr>
        <tr>
          <td    height="26"    colspan="2"    align="center"    valign="middle"
bgcolor="#CCCCCC">
    <input type="submit" name="tijiao" value="提交">    
    <input type="reset" name="Submit2" value="重 置"></td>
          </tr>
      </table>
    </form>
    <?php
    }
    ?>
    </td>
     </tr>
     <tr>
      <td height="1" bgcolor="#CCCCCC"></td>
     </tr>
    </table>
    <?php
    include "inc/foot.php";
```

```
?>
</body>
</html>
```

发布新帖的页面效果如图 21-5 所示。



图 21-5　发布新帖页面

文件 note_show.php 位于配套素材包的 ch21\下，是显示帖子和相关回复的页面。具体代码如下。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>===迅捷 BBS 系统===</title>
<link href="inc/style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<?php
include "inc/mysql.inc";
include "inc/myfunction.inc";
include "inc/head.php";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
include "inc/total_info.php";
$module_id=$_GET[module_id];
$note_id=$_GET[note_id];
?>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="73%" height="30"><a href="./">迅捷B B S 系统</a>>>
```

```
<?php
echo "<a href=module_list.php?module_id=".$module_id.">";
echo $bb->son_module_idtofather_name($module_id);
echo "</a>>>";
echo $bb->son_module_idtomodule_name($module_id);
//删除回复
$del_id=$_GET[del_id];
if ($del_id!=""){
    if ($bb->son_module_idtouser_name($module_id)==$_SESSION[user_name]){
    $del_query="delete from note_info where id='$del_id'";
    $aa->excu($del_query);
    echo "<br />===删除回复成功！===";
    }
}
//添加回复
$tijiao=$_POST[tijiao];
if ($tijiao=="提 交"){
    $title=$_POST[title];
    $cont=$_POST[cont];
    $cont=$bb->str_to($cont);
    $today=date("Y-m-d H:i:s");
    if ($_SESSION[user_name]==""){
        $user_name="游客";
    }else{
        $user_name=$_SESSION[user_name];
    }
    $query="insert into note_info(up_id,title,cont,time,user_name)
values('$note_id','$title','$cont','$today','$user_name')";
    $aa->excu($query);
}
?></td>
   <td width="16%" align="right" valign="middle"><a href="#huifu">
img src="pic_sys/reply.gif" width="82" height="20"></a></td>
   <td width="11%" align="right" valign="middle"><a href="new_note.php">
<img src="pic_sys/post.gif" width="82" height="20"></a></td>
  </tr>
</table>
<?php
$query="select * from note_info where up_id='$note_id' order by time";
$add="module_id=".$module_id."&note_id=".$note_id."&";
?>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
   <td height="30" align="left" valign="middle"><?php $bb->page($query,
$page_id,$add,20)?>
```

```
    </td>
      </tr>
    </table>
    <?php
    $query2="select * from note_info where id='$note_id'";
    $result2=$aa->excu($query2);
    $note2=mysqli_fetch_array($result2);
    ?>
    <table width="98%" border="0" align="center" cellpadding="0" cellspacing="1"
bgcolor="#FFFFFF">
      <tr>
        <td width="71%" height="25" align="left" valign="middle" bgcolor="5F8AC5">
标题:
    <?php
    echo $note2[title]?></td>
        <td width="29%" align="center" valign="middle" bgcolor="5F8AC5">发帖时间:
    <?php echo $note2[time]?></td>
      </tr>
      <tr>
        <td height="1" colspan="2" bgcolor="#CCCCCC"></td>
      </tr>
    </table>
    <table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
      <tr>
        <td     width="16%"     height="15"     align="center"     valign="top"><img
src="pic_sys/head2.jpg"
    width="85" height="90"><br /><?php echo $note2[user_name]?></td>
        <td align="left" valign="middle"><?php echo $note2[cont]?></td>
      </tr>
      <tr>
        <td height="8" colspan="2" align="center" valign="top" bgcolor="#5F8AC5">
</td>
      </tr>
    </table>
    <?php
    $rst=$aa->excu($query);
    if (mysqli_num_rows ($rst)!=0){
    ?>
    <table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
      <?php
      while ($note=mysqli_fetch_array($rst)){
      ?>
      <tr>
        <td width="16%" height="120" rowspan="3" align="center" valign="top">
     <?php
```

```
    if ($note[user_name]=="游客"){
    ?>
    <img src="pic_sys/head1.png" width="100" height="100">
    <br />
    游客
    <?php
    }else{
    ?>
    <img src="pic_sys/head2.jpg" width="85" height="90"><br />
    <?php
    echo $note[user_name];
    }
    ?>
    </td>
       <td  width="54%"  height="26"  align="left"  valign="middle"><?php  echo
$note[title]?></td>
       <td  width="18%"  height="26"  align="center"  valign="middle"><?php  echo
$note[time]?></td>
       <td width="12%" height="26" align="center" valign="middle">
    <?php
    if    ($bb->son_module_idtouser_name($module_id)==$_SESSION[user_name]    ||
$_SESSION
    [manage_tag]==1){
    echo "<a href=?
    module_id=".$module_id."&note_id=".$note_id."&page_id=".$page_id."&del_id="
.$note[id].">
    删除回复</a>";
    }
    ?>  </td>
     </tr>
     <tr>
       <td height="1" colspan="3" align="left" valign="top" bgcolor="#CCCCCC">
</td>
     </tr>
       <tr>
       <td   height="70"   colspan="3"   align="left"   valign="top"><?php   echo
$note[cont]?></td>
     </tr>
     <tr>
       <td height="2" colspan="4" align="center" valign="top" bgcolor="#CCCCCC">
</td>
     </tr>
     <?php }?>
    </table>
    <?php
```

```
    }
   ?>
   <table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
     <tr>
       <td height="30" align="right" valign="bottom"><?php
    $query="select * from note_info where up_id='$note_id' order by time desc";
    $bb->page($query,$page_id,$add,20);
    ?></td>
     </tr>
   </table>
   <form name="form1" method="post" action="#">
   <table width="500" border="0" align="center" cellpadding="0" cellspacing="2">
     <tr>
       <td    height="26"    colspan="2"    align="center"    valign="middle"
bgcolor="#CCCCCC">
   <a name="huifu">回 复 此 帖</a></td>
     </tr>
     <tr>
       <td width="122" height="26" align="right" valign="middle" bgcolor=
"#CCCCCC">标题:</td>
       <td width="372" height="26" align="left" valign="middle" bgcolor=
"#CCCCCC">

    <?php
        $reply_title="回复:".$bb->note_idtotitle($note_id);
        echo $reply_title;
    ?>
    <input type="hidden" name="title" value="<?php echo $reply_title?>">
    </td>
     </tr>
     <tr>
       <td height="26" align="right" valign="middle" bgcolor="#CCCCCC">内容:</td>
       <td height="26" align="left" valign="middle" bgcolor="#CCCCCC">
   <textarea name="cont" cols="50" rows="8"></textarea></td>
     </tr>
     <tr>
       <td  height="26"  align="right"  valign="middle"  bgcolor="#CCCCCC"> 发 帖
人:</td>
       <td height="26" align="left" valign="middle" bgcolor="#CCCCCC"> 
    <?php
        if ($_SESSION[user_name]==""){
            echo "游客";
        }else{
            echo $_SESSION[user_name];
        }
```

```
 ?></td>
  </tr>
  <tr>
    <td height="26" align="right" valign="middle" bgcolor="#CCCCCC">时间:</td>
    <td height="26" align="left" valign="middle" bgcolor="#CCCCCC">系统将自动
记录! </td>
  </tr>
  <tr>
    <td     height="26"     colspan="2"     align="center"     valign="middle"
bgcolor="#CCCCCC">
<input type="submit" name="tijiao" value="提 交">
          
      <input type="reset" name="Submit2" value="重 置"></td>
  </tr>
</table>
</form>
<?php
include "inc/foot.php";
?>
</body>
</html>
```

回复帖子的页面效果如图 21-6 所示。



图 21-6 回复帖子页面

文件 module_list.php 位于配套素材包的 ch21\下，是显示子模块下帖子列表的页面。具体代码如下。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>===迅捷 BBS 系统===</title>
```

```
<link href="inc/style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<?php
@session_start();
include "inc/mysql.inc";
include "inc/myfunction.inc";
include "inc/head.php";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
include "inc/total_info.php";
$module_id=@$_GET['module_id'];
$del_id=@$_GET['del_id'];
if ($bb->son_module_idtouser_name($module_id)==@$_SESSION['user_name']){
 $query="delete from note_info where id='".$del_id."'";
 $aa->excu($query);
 }
$query="select * from note_info where module_id='".$module_id."' order by time
desc";
$add="module_id=".$module_id."&";
?>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="73%" height="30"><a href="./">迅捷 BBS 系统</a>>>
 <?php
 echo "<a href=module_list.php?module_id=".$module_id.">";
 echo $bb->son_module_idtofather_name($module_id);
 echo "</a>>>";
 echo $bb->son_module_idtomodule_name($module_id);
 ?></td>
    <td width="27%" align="right" valign="middle"><a href="new_note.php">
<img src="pic_sys/post.gif" width="82" height="20"></a></td>
  </tr>
</table>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td height="30" align="left" valign="middle"><?php $bb->page($query,
@$page_id,$add,20)?></td>
  </tr>
</table>
<table width="98%" border="0" align="center" cellpadding="0" cellspacing="1"
bgcolor="#FFFFFF">
  <tr>
    <td width="3%" height="25" align="center" valign="middle" bgcolor="5F8AC5"
```

```
> </td>
        <td width="5%" align="center" valign="middle" bgcolor="5F8AC5">人气</td>
        <td width="50%" align="center" valign="middle" bgcolor="5F8AC5">标    题
</td>
        <td width="9%" align="center" valign="middle" bgcolor="5F8AC5">发起人</td>
        <td width="5%" align="center" valign="middle" bgcolor="5F8AC5">帖子数</td>
        <td width="16%" align="center" valign="middle" bgcolor="5F8AC5">最后发表时
间</td>
        <td width="12%" align="center" valign="middle" bgcolor="5F8AC5">最后发表人
</td>
      </tr>
      <?php
     $result=$aa->excu($query);
     while($note=mysqli_fetch_array($result)){
      ?>
      <tr>
        <td height="25" align="center" valign="middle"><img src="pic_sys/li-3.gif"
    width="14" height="11"></td>
        <td height="25" align="center" valign="middle"><?php echo $bb->note_
idtotimes($note['id']);?></td>
        <td height="25" align="left" valign="middle"><?php
     echo "<a href=note_show.php?module_id=".$module_id."&note_id=".$note['id'].
">".$note['title']."</a>";
     if ($bb->son_module_idtouser_name($module_id)==$_SESSION['user_name']
    || $_SESSION['manage_tag']==1){
     echo "  
    <a
href=module_list.php?module_id=".$module_id."&page_id=".$page_id."&del_id=".$n
ote['id'].">
    删除此帖</a>";
     }
     ?></td>
        <td height="25" align="center" valign="middle"><?php echo $note['user_
name'];?></td>
        <td height="25" align="center" valign="middle">
    <?php echo $bb->note_idtonote_num($note['id']);?></td>
        <td height="25" align="center" valign="middle">
    <?php echo $bb->note_idtolast_time($note['id']);?></td>
        <td height="25" align="center" valign="middle">
    <?php echo $bb->note_idtolast_user_name($note['id']);?></td>
      </tr>
      <tr>
        <td height="1" colspan="7" bgcolor="#CCCCCC"></td>
      </tr>
      <?php
```

```
    }
    ?>
  </table>
  <table width="98%" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr>
      <td height="30" align="right" valign="bottom">
   <?php
   $query="select * from note_info where module_id='".$module_id."' order by time
desc";
   $bb->page($query,@$page_id,$add,20)
   ?></td>
    </tr>
  </table>
  <?php
  include "inc/foot.php";
  ?>
  </body>
  </html>
```

显示帖子的页面效果如图 21-7 所示。

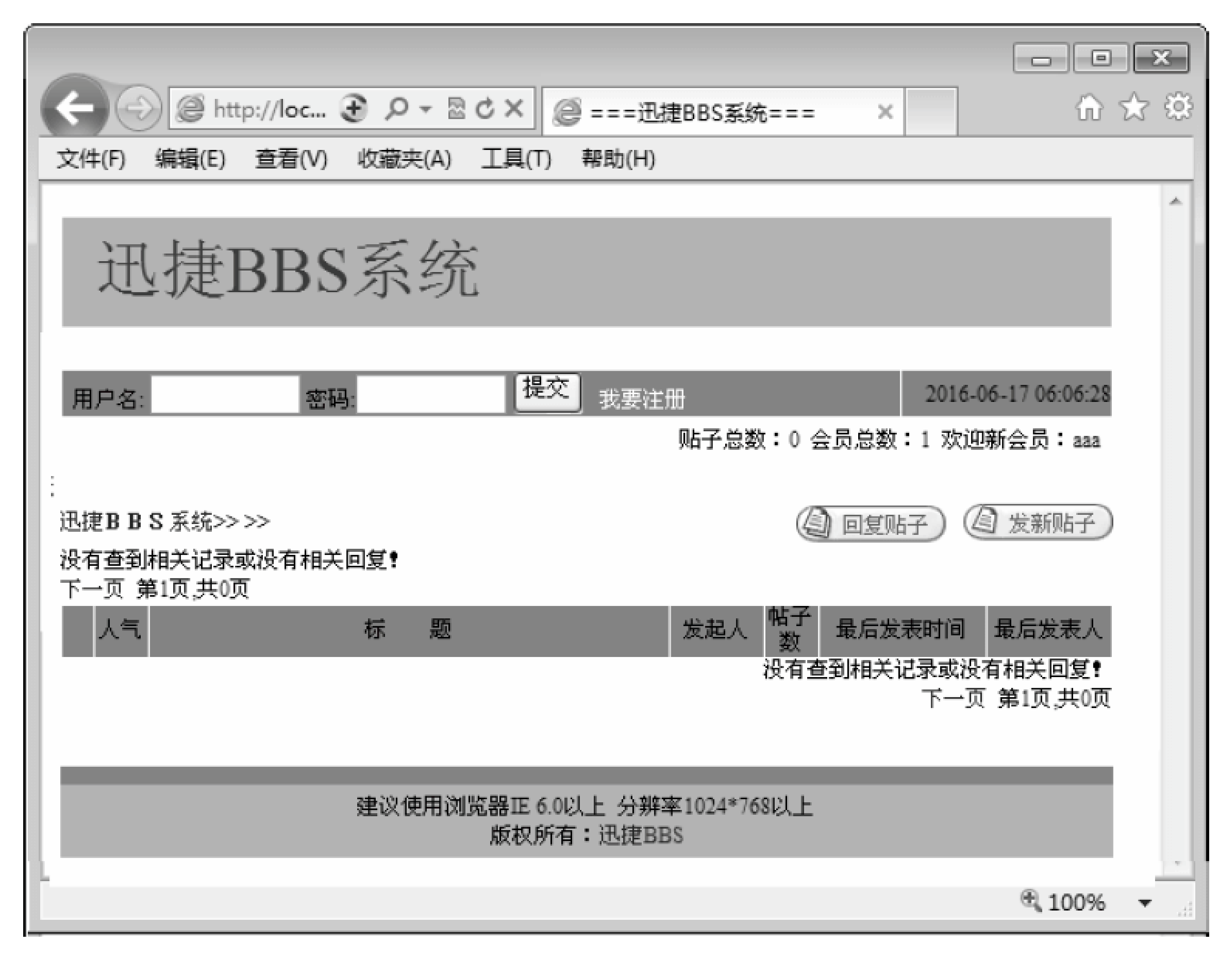


图 21-7　显示帖子列表页面

21.3.5　后台管理系统的相关页面

下面介绍后台管理系统的相关页面。

文件 login.php 位于配套素材包的 ch21\manage\下，是管理用户的登录页面。具体代码如下。

```
  <?php
     include "../inc/mysql.inc";
```

```
include "../inc/myfunction.inc";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
   $_SESSION['manage_name']="";
$_SESSION['manage_tag']="";
?>
<head>
<style>
<!--
td          { font-size: 10pt }
-->
</style>
<title>:::管理员登录==迅捷 BBS 系统:::</title>
</head>
<body  onLoad="tijiao.username.value='';tijiao.username.focus();">

<p align="center">  </p>
<br />
  </font></p>
<div align="center">
  <center>

  <form method=POST name="tijiao" action="auth.php">
  <table  border="1"  cellpadding="0"  cellspacing="0"  bordercolor="#111111"
width="240"
    height="126"      bordercolorlight="#FFFFFF"      bordercolordark="#FFFFFF"
style="border-collapse: collapse">
      <tr>
        <td width="238" colspan="2" height="25" bgcolor="#A8A3AD">
        <p  align="center"><b><font  color="#FFFFFF"> 迅 捷  BBS  后 台 管 理 系 统
</font></b></td>
      </tr>
      <tr>
        <td width="64" height="26" bgcolor="#E3E1E6">
        <p align="center">账  号：</td>
        <td height="26" bgcolor="#E3E1E6" width="173">
        <input  type="text"  name="username"  size="20"  style="color:  #A8A3AD;
border-style: solid;
    border-width:  1;  padding-left:  4;  padding-right:  4;  padding-top:  1;
padding-bottom: 1"></td>
      </tr>
      <tr>
        <td width="64" height="26" bgcolor="#E3E1E6">
        <p align="center">口  令：</td>
```

```
        <td height="26" bgcolor="#E3E1E6" width="173">
        <input type="password" name="password" size="20" style="color: #A8A3AD;
border-style: solid;
    border-width:  1;  padding-left:  4;  padding-right:  4;  padding-top:  1;
padding-bottom: 1"></td>
      </tr>
      <tr>
        <td width="238" height="29" bgcolor="#E3E1E6" colspan="2">
        <p align="center"><input type="submit" value="登 录" name="login">
    
        <input type="reset" value="取 消" name="cancel"></td>
      </tr>
      <tr>
        <td width="238" colspan="2" height="20" bgcolor="#A8A3AD">  </td>
      </tr>
     </table>
     </form>
     </center>
   </div>
   </body>
```

管理用户的登录页面效果如图 21-8 所示。



图 21-8　管理用户的登录页面

文件 session.inc 位于配套素材包的 ch21\manage\下，是检验 Session 是否存在的页面。具体代码如下。

```
<?php
@session_start();
//if ($_SESSION['manage_name']=="" and $_SESSION['manage_tag']!=1){
//    header("location:./login.php");
//    }
?>
```

文件 index_top.php 位于配套素材包的 ch21\manage\下，是后台管理主页面的上部页面。具体代码如下。

```
<HTML>
<HEAD><TITLE>顶部管理导航菜单</TITLE>
<META http-equiv=Content-Type content="text/html; charset=gb2312">
<STYLE type=text/css>A:link {
 COLOR: #ffffff; TEXT-DECORATION: none
}
A:hover {
 COLOR: #ffffff
}
A:visited {
 COLOR: #f0f0f0; TEXT-DECORATION: none
}
.spa {
 FONT-SIZE: 9pt; FILTER: Glow(Color=#0F42A6, Strength=2) dropshadow(Color=#0F42A6, OffX=2, OffY=1,); COLOR: #8aade9; FONT-FAMILY: '宋体'
}
IMG {
 FILTER: Alpha(opacity:100); chroma: #FFFFFF)
}
</STYLE>
<SCRIPT language=JavaScript type=text/JavaScript>
function preloadImg(src) {
  var img=new Image();
  img.src=src
}
preloadImg('image/admin_top_open.gif');

var displayBar=true;
function switchBar(obj) {
  if (displayBar) {
    parent.frame.cols='0,*';
    displayBar=false;
    obj.src='image/admin_top_open.gif';
    obj.title='打开左边管理导航菜单';
  } else {
    parent.frame.cols='200,*';
    displayBar=true;
    obj.src='image/admin_top_close.gif';
    obj.title='关闭左边管理导航菜单';
  }
}
</SCRIPT>
```

```
<META content="MSHTML 6.00.2900.2963" name=GENERATOR></HEAD>
<BODY leftMargin=0 background=image/admin_top_bg.gif
topMargin=0>
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
  <TBODY>
  <TR vAlign=center>
    <TD width=60><IMG title=关闭左边管理导航菜单 style="CURSOR: hand"
      onclick=switchBar(this)
src="image/admin_top_close.gif"></TD>
    <TD width=92><A href="login.php" target="_parent" ><IMG
      src="image/top_an_1.gif" border=0></A></TD>
    <TD width=92><A href="#"></A></TD>
    <TD width=104><A href="#"></A></TD>
    <TD width=92><A href="#"></A></TD>
    <TD width=92><A href="#"></A></TD>
    <TD class=spa align=right>ET PHP SOUND CODE DEVELOP  
  </TD>
  </TR>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

文件 index_left.php 位于配套素材包的 ch21\manage\下，是后台管理主页面的左侧页面。具体代码如下。

```
<?php
include "session.inc";
?>
<HTML>
<HEAD><TITLE>管理导航菜单</TITLE>
<META http-equiv=Content-Type content="text/html; charset=gb2312">
<SCRIPT src="menu.js"></SCRIPT>
<LINK href="left.css" type=text/css rel=stylesheet>
</HEAD>
<BODY leftMargin=0 topMargin=0 marginwidth="0" marginheight="0">
<TABLE cellSpacing=0 cellPadding=0 width=180 align=center border=0>
  <TBODY>
  <TR>
    <TD                      vAlign=top                      height=44><IMG
src="image/title.gif"></TD></TR></TBODY></TABLE>
<TABLE cellSpacing=0 cellPadding=0 width=180 align=center>
  <TBODY>
  <TR>
    <TD class=menu_title id=menuTitle0
    onmouseover="this.className='menu_title2';"
```

```
        onmouseout="this.className='menu_title';"
        background=image/title_bg_quit.gif
          height=26>  <A href="Admin_Main.php"  target="_top"><B><SPAN
class=glow>
    管理首页</SPAN></B></A><SPAN class=glow> |
          </SPAN><A href="login.php" target="_top"><B><SPAN class=glow>退出
    </SPAN></B></A> </TD></TR>
      <TR>
        <TD id=submenu0 background=image/title_bg_admin.gif height=97>
          <DIV style="WIDTH: 180px">
          <TABLE cellSpacing=0 cellPadding=0 width=130 align=center>
            <TBODY>
            <TR>
              <TD height=16>您的用户名：<?php echo @$_SESSION['manage_name'];?>
</TD></TR>
            <TR>
              <TD height=16>您的身份：<?php echo @$_SESSION['manage_name'];?>
</TD></TR>
           <TR>
            <TD height=16>IP：<?php echo $_SERVER["REMOTE_ADDR"];?></TD>
              </TR>
            <TR>
         <TD height=16> </TD>
              </TR>
         </TBODY></TABLE></DIV>
          <DIV style="WIDTH: 167px">
          <TABLE cellSpacing=0 cellPadding=0 width=130 align=center>
            <TBODY>
            <TR>
              <TD
height=20></TD></TR></TBODY></TABLE></DIV></TD></TR></TBODY></TABLE>
    <TABLE cellSpacing=0 cellPadding=0 width=167 align=center>
      <TBODY>
      <TR>
        <TD class=menu_title id=menuTitle1
        onmouseover="this.className='menu_title2'" style="CURSOR: hand"
        onclick="new Element.toggle('submenu1')"
        onmouseout="this.className='menu_title'"
        background=image/Admin_left_1.gif height=28 ;><SPAN class=glow>论坛版块管
理</SPAN></TD>
      </TR>
      <TR>
        <TD id=submenu1 style="DISPLAY: none" align=right>
          <DIV class=sec_menu style="WIDTH: 165px">
          <TABLE cellSpacing=0 cellPadding=0 width=132 align=center>
```

```
            <TBODY>
         <TR>
              <TD height=20><A href="father_module_add.php" target=main>父版块添加
</A></TD>
         </TR>
         <TR>
              <TD height=20><A href="father_module_list.php" target=main>父版块管理
</A></TD>
         </TR>
         <TR>
              <TD  height=20><A  href="son_module_add.php"  target=main> 子 版 块 添 加
</A></TD>
         </TR>
            <TR>
              <TD  height=20><A  href="son_module_list.php"  target=main>子 版 块 管 理
</A></TD>
            </TR>
              </TABLE>
          </DIV>
          <DIV style="WIDTH: 158px">
          <TABLE cellSpacing=0 cellPadding=0 width=130 align=center>
            <TBODY>
            <TR>
              <TD
height=4></TD></TR></TBODY></TABLE></DIV></TD></TR></TBODY></TABLE>
    <TABLE cellSpacing=0 cellPadding=0 width=167 align=center>
      <TBODY>
      <TR>
        <TD class=menu_title id=menuTitle2
        onmouseover="this.className='menu_title2'" style="CURSOR: hand"
        onclick="new Element.toggle('submenu2')"
        onmouseout="this.className='menu_title'"
        background=image/Admin_left_11.gif height=28 ;><SPAN class=glow>论坛用户管
理</SPAN></TD>
      </TR>
      <TR>
        <TD id=submenu2 style="DISPLAY: none" align=right>
          <DIV class=sec_menu style="WIDTH: 165px">
          <TABLE cellSpacing=0 cellPadding=0 width=132 align=center>
            <TBODY>
            <TR>
              <TD height=20><A href="user_list.php" target=main>所有用户</A></TD>
            </TR>
         <TR>
              <TD height=20><A href="user_js.php" target=main>用户检索</A></TD>
```

```
        </TR>
    </TBODY></TABLE>
          </DIV>
          <DIV style="WIDTH: 158px">
          <TABLE cellSpacing=0 cellPadding=0 width=130 align=center>
            <TBODY>
            <TR>
              <TD
height=4></TD></TR></TBODY></TABLE></DIV></TD></TR></TBODY></TABLE>
    <TABLE cellSpacing=0 cellPadding=0 width=167 align=center>
      <TBODY>
      <TR>
        <TD class=menu_title id=menuTitle3
        onmouseover="this.className='menu_title2'" style="CURSOR: hand"
        onclick="new Element.toggle('submenu3')"
        onmouseout="this.className='menu_title'"
        background=image/Admin_left_3.gif height=28 ;><SPAN class=glow>安全管理
</SPAN></TD>
      </TR>
      <TR>
        <TD id=submenu3 style="DISPLAY: none" align=right>
          <DIV class=sec_menu style="WIDTH: 165px">
          <TABLE cellSpacing=0 cellPadding=0 width=132 align=center>
            <TBODY>
         <TR>
              <TD  height=20><A  href="user_pw_change.php"  target=main> 密 码 更 改
</A></TD>
         </TR>
         <TR>
              <TD height=20><A href="../" target=_blank>帖子管理</A></TD>
         </TR>
    </TBODY></TABLE>
          </DIV>
          <DIV style="WIDTH: 158px">
          <TABLE cellSpacing=0 cellPadding=0 width=130 align=center>
            <TBODY>
            <TR>
              <TD
height=4></TD></TR></TBODY></TABLE></DIV></TD></TR></TBODY></TABLE>
    <TABLE cellSpacing=0 cellPadding=0 width=167 align=center>
      <TBODY>
      <TR>
        <TD class=menu_title id=menuTitle208
        onmouseover="this.className='menu_title2';"
        onmouseout="this.className='menu_title';"
```

```
        background=image/Admin_left_04.gif
          height=28><SPAN>系统信息</SPAN> </TD></TR>
      <TR>
        <TD align=right>
          <DIV class=sec_menu style="WIDTH: 165px">
          <TABLE cellSpacing=0 cellPadding=0 width=130 align=center>
            <TBODY>
            <TR>
              <TD height=20><br />版本号:verson 1.0<br />版权所有:  <A href=
"http://www.quickbbs.net/" target=_blank>迅捷BBS</A>
           <br />设计制作:  <A href="http://www.etpt.net/" target=_blank>迅捷
BBS</A>
           <br />技术支持:  <A href="http://bbs.etpt.net/" target=_blank>迅捷
BBS</A>
           <br /><br /></TD></TR></TBODY></TABLE></DIV></TD></TR>
    </TBODY></TABLE>
    </BODY>
    </HTML>
```

文件 index_right.php 位于配套素材包的 ch21\manage\下，是后台管理主页面的右侧页面。具体代码如下。

```
    <?php
    include "fun_head.php";
    head("管理首页");
    ?>
    <TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
      <TBODY>
      <TR>
        <TD width=20 rowSpan=2> </TD>
        <TD class=topbg align=middle width=100>论坛版块管理</TD>
        <TD width=300> </TD>
        <TD width=40 rowSpan=2> </TD>
        <TD class=topbg align=middle width=100>论坛用户管理</TD>
        <TD width=300> </TD>
        <TD width=21 rowSpan=2> </TD></TR>
      <TR class=topbg2>
        <TD colSpan=2 height=1></TD>
        <TD colSpan=2></TD></TR></TBODY></TABLE>
    <TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
      <TBODY>
      <TR>
        <TD width=20> </TD>
        <TD width=400><br />
      本管理版块共分四个子模块:父版块添加、父版块管理、子版块添加、子版块管理。
    其中父版块添加和子版块添加可以实现本论坛的父版块和子版块的添加;
```

```
父版块的管理和子版块的管理可以实现本论坛父版块和子版块的删除、编辑等功能。<br />
          </TD>
    <TD width=40> </TD>
    <TD
      width=400><br />
本管理模块共分两个子版块:所有用户和用户检索。
其中所有用户按用户注册的先后顺序分页依次列出。
用户检索是由管理员输入要查询的论坛注册用户的用户名，系统通过数据库查询出该用户的
相关信息，并列出，而且管理员也可以在查询注册用户后直接删除该用户。<br />
      <br /></TD>
    <TD width=21> </TD></TR></TBODY></TABLE>
<TABLE height=10 cellSpacing=0 cellPadding=0 width="100%" border=0>
  <TBODY>
  <TR>
    <TD></TD></TR></TBODY></TABLE>
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
  <TBODY>
  <TR>
    <TD width=20 rowSpan=2> </TD>
    <TD class=topbg align=middle width=100>安全管理</TD>
    <TD width=300> </TD>
    <TD width=40 rowSpan=2> </TD>
    <TD class=topbg align=middle width=100>系统信息</TD>
    <TD width=300> </TD>
    <TD width=21 rowSpan=2> </TD></TR>
  <TR class=topbg2>
    <TD colSpan=2 height=1></TD>
    <TD colSpan=2></TD></TR></TBODY></TABLE>
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
  <TBODY>
  <TR>
    <TD width=20> </TD>
    <TD
      width=400><br />
本管理模块共分两个子版块:密码更改和帖子管理。
其中密码更改提供了管理员更改密码的功能，但必须输入原密码和两次新密码;
帖子管理是直接进入论坛的主界面，但与普通注册用户所不同的是，
在论坛的每个发帖和回复后面都多了一个删除按钮，可以通过此按钮删除相关帖子或回复。</TD>
    <TD width=40> </TD>
    <TD
      width=400 valign="top"><br />
      本版块提供了本论坛的版本号、版权所有、设计制作以及技术支持等信息。</TD>
    <TD width=21> </TD></TR></TBODY></TABLE>
<TABLE height=70 cellSpacing=0 cellPadding=0 width="100%" border=0>
  <TBODY>
```

```
    <TR>
  <TD></TD></TR></TBODY></TABLE>
  <TABLE height=10 cellSpacing=0 cellPadding=0 width="100%" border=0>
    <TBODY>
    <TR>
      <TD></TD></TR></TBODY></TABLE>
  <br />
   <?php
   include "bottom.php";
   ?>
```

文件 bootom.php 位于配套素材包的 ch21\manage\下，是后台管理主页面的版权页面。具体代码如下。

```
  <TABLE class=border cellSpacing=1 cellPadding=2 width="100%" align=center
  border=0>
    <TBODY>
    <TR align=middle>
      <TD class=topbg height=25><SPAN class=Glow>Copyright 2012 &copy;
   <a href="http://www.quickbbs.net" target="_blank"><font color="#FFFFFF">迅捷
BBS</font></a>
        All Rights Reserved.</SPAN> </TD></TR>
  </TBODY>
  </TABLE>
```

文件 index.php 位于配套素材包的 ch21\manage\下，是后台管理的主页面。具体代码如下。

```
  <?php
  include "session.inc";
  ?>
  <HTML>
  <HEAD>
  <TITLE>===迅捷 BBS 系统-后台管理===</TITLE>
  <META http-equiv=Content-Type content="text/html; charset=gb2312">
  </HEAD>
  <FRAMESET id=frame border=false frameSpacing=0 rows=* frameBorder=0 cols=200,*
scrolling="yes">
      <FRAME  name=left  marginWidth=0  marginHeight=0  src="index_left.php"
scrolling=yes>
      <FRAMESET  border=false  frameSpacing=0  rows=53,*  frameBorder=0  cols=*
scrolling="yes">
          <FRAME name=top src="index_top.php" scrolling=no>
          <FRAME name=main src="index_right.php">
      </FRAMESET>
  </FRAMESET>
```

文件 fun_head.php 位于配套素材包的 ch21\manage\下，是后台管理中右侧页面中的头文件。具体代码如下。

```
<?php
 function head($str){
?>
<LINK href="Admin_Style.css" rel=stylesheet>
<STYLE type=text/css>
.STYLE4 {
 COLOR: #000000
}
</STYLE>
<BODY leftMargin=0 topMargin=0 marginheight="0" marginwidth="0">
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
  <TBODY>
  <TR>
    <TD width=392 rowSpan=2><img height=126
      src="image/adminmain01.gif" width=392></TD>
    <TD vAlign=top background=image/adminmain0line2.gif
    height=114>
      <TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
        <TBODY>
        <TR>
          <TD height=20></TD></TR>
        <TR>
          <TD><SPAN class=STYLE4>频道管理中心</SPAN></TD></TR>
        <TR>
          <TD height=8><IMG height=1
            src="image/adminmain0line.gif" width=283></TD></TR>
        <TR>
          <TD><IMG src="image/img_u.gif"
            align=absMiddle>欢迎进入管理<FONT
        color=#ff0000></FONT></TD></TR>
        <TR>
          <TD><IMG src="image/img_u.gif" align=absMiddle>
          <?php
          echo "当前位置：<b><font color=#ffffff>".$str."</font></b>";
          ?></TD>
        </TR>
        </TBODY></TABLE></TD></TR>
  <TR>
    <TD vAlign=bottom background=image/adminmain03.gif
    height=9><IMG height=12 src="image/adminmain02.gif"
      width=23></TD></TR></TBODY></TABLE>
   <?php
```

```
    }
?>
```

文件 father_module_add.php 位于配套素材包的 ch21\manage\下，是父版块添加页面。具体代码如下。

```
<?php
include "session.inc";
include "fun_head.php";
head("父版块添加");
include "../inc/mysql.inc";
include "../inc/myfunction.inc";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
$add_tag=@$_GET['add_tag'];
if ($add_tag==1){
$show_order=$_POST['show_order'];
$module_name=$_POST['module_name'];
 if ($show_order=="" or $module_name==""){
 echo "===对不起，您添加父版块不成功：
<font color=red>显示序号和父版块名称全不能为空</font>! ===";
 }else{
 $query="insert into
father_module_info(module_name,show_order)
values('$module_name','$show_order')";
 $aa->excu($query);
 echo "===恭喜您，添加父版块添加成功！===";
 }
}
?>
<table width="100%" height="389" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
   <tbody>
     <tr>
       <td width="20"> </td>
       <td valign="top"><br />
         <form action="?add_tag=1" method="post" name="form1" id="form1">
           <table   width="408"   height="87"   border="0"   align="center"
cellpadding="0" cellspacing="1" bordercolor="#FFFFFF" bgcolor="449ae8">
             <tr bgcolor="#dcccccc">
               <td width="94" height="25" bgcolor="e0eef5"><div align="right">
显示序号:</div></td>
               <td  width="306"  bgcolor="e0eef5"><input  type="text"  size="6"
name="show_order" />
                 请填写一整数，如：1
```

```
                 。</td>
               </tr>
               <tr bgcolor="#dddddd">
                 <td  height="25"  bgcolor="#FFFFFF"><div  align="right">父版块名
称:</div></td>
                 <td      bgcolor="#FFFFFF"><input      type="text"      size="20"
name="module_name" /></td>
               </tr>
               <tr bgcolor="#dddddd">
                 <td height="33" colspan="2" bgcolor="e0eef5"><div align="center">
                     <input name="submit" type="submit" value="提交" />
                         
                   <input name="reset" type="reset" value="重置" />
                 </div></td>
               </tr>
             </table>
           </form>
           <br />
         <br /></td>
         <td width="20"> </td>
       </tr>
     </tbody>
   </table>
   <?php
    include "bottom.php";
   ?>
```

文件 father_module_bj.php 位于配套素材包的 ch21\manage\下，是父版块编辑页面。具体代码如下。

```
   <?php
   include "session.inc";
   include "fun_head.php";
   head("父版块管理==>>编辑");
   include "../inc/mysql.inc";
   include "../inc/myfunction.inc";
   $aa=new mysqlconn;
   $bb=new myfunction;
   $aa->link("");
   $module_id=$_GET['module_id'];
   $update_tag=@$_GET['update_tag'];
   if ($update_tag==1){
   $show_order=$_POST['show_order'];
   $module_name=$_POST['module_name'];
    if ($show_order=="" or $module_name==""){
    echo "===对不起，您编辑父版块不成功：<font color=red>
```

```
    显示序号和父版块名称全不能为空</font>! ===";
     }else{
     $query="update father_module_info set module_name='$module_name',show_order=
'$show_order
    ' where id='$module_id'";
     $aa->excu($query);
     echo "===恭喜您，添加父版块添加成功! ===";
     }
    }
    $query="select * from father_module_info where id='$module_id'";
    $rst=$aa->excu($query);
    $module=mysqli_fetch_array($rst,MYSQL_ASSOC);
    ?>
    <table width="100%" height="389" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
      <tbody>
        <tr>
          <td width="20"> </td>
          <td valign="top"><br />
            <form action="?update_tag=1&module_id=<?php echo $module_id?>"
    method="post" name="form1" id="form1">
              <table width="408" height="87" border="0" align="center" cellpadding="0"
    cellspacing="1" bordercolor="#FFFFFF" bgcolor="449ae8">
                <tr bgcolor="#dcccccc">
                  <td width="94" height="25" bgcolor="e0eef5"><div align="right">
显示序号:</div></td>
                  <td width="306" bgcolor="e0eef5"><input type="text" size="6"
    name="show_order" value="<?php echo $module['show_order']?>" />
                    请填写一整数，如：1
                  。</td>
                </tr>
                <tr bgcolor="#dddddd">
                  <td  height="25"  bgcolor="#FFFFFF"><div  align="right"> 父 版 块 名
称:</div></td>
                  <td bgcolor="#FFFFFF"><input type="text" size="20" name="module_name"
    value="<?php echo $module['module_name']?>" /></td>
                </tr>
                <tr bgcolor="#dddddd">
                  <td height="33" colspan="2" bgcolor="e0eef5"><div align="center">
                      <input name="submit" type="submit" value="提交" />
                          
                    <input name="reset" type="reset" value="重置" />
                  </div></td>
                </tr>
              </table>
```

```
        </form>
        <br />
      <br /></td>
      <td width="20"> </td>
    </tr>
  </tbody>
</table>
<?php
 include "bottom.php";
?>
```

文件 father_module_list.php 位于配套素材包的 ch21\manage\下，是父版块添加显示页面。具体代码如下。

```
<?php
include "session.inc";
include "fun_head.php";
head("父版块管理");
include "../inc/mysql.inc";
include "../inc/myfunction.inc";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
///////////删除父版块////////////////////////////////
$del_tag=@$_GET['del_tag'];
if ($del_tag==1){
 $module_id=$_GET['module_id'];
 $query="delete from father_module_info where id='$module_id'";
 $aa->excu($query);
 echo "==恭喜您，删除父版块信息成功！==<br />";
}
///////////按显示顺序查询父版块信息表/////////////////////
 $query="select * from father_module_info order by show_order";
 $rst=$aa->excu($query);
?>
<table width="100%" height="390" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
  <tbody>
    <tr>
      <td width="20"> </td>
      <td valign="top"><br /><table width="80%" border="0" align="center"
cellpadding="0" cellspacing="1" bgcolor="449ae8">
        <tr bgcolor="#cccccc">
          <td width="92" height="23" bgcolor="e0eef5"><div align="center">编号
</div></td>
          <td width="193" bgcolor="e0eef5"><div align="center"> 显 示 序 号
```

```
</div></td>
              <td width="368" bgcolor="e0eef5"><div align="center"> 父 版 块 名 称
</div></td>
              <td colspan="2" bgcolor="e0eef5"><div align="center">操作</div></td>
            </tr>
    <?php
     $m=0;
     while($module=mysqli_fetch_array($rst,MYSQL_ASSOC)){
     $m++;
    ?>
            <tr>
              <td height="19" bgcolor="#FFFFFF"><div align="center"><?php echo
$m;?></div></td>
              <td     bgcolor="#FFFFFF"><div     align="center"><?php     echo
$module['show_order']?></div></td>
              <td     bgcolor="#FFFFFF"><div     align="center"><?php     echo
$module['module_name']?></div></td>
              <td width="134" align="center"
    bgcolor="#FFFFFF"><a href="father_module_bj.php?module_id=
    <?php echo $module['id'];?>">编辑</a></td>
              <td     width="142"     align="center"     bgcolor="#FFFFFF"><a
href="?del_tag=1&module_id=
    <?php echo $module['id']?>">删除</a></td>
            </tr>
            <?php }?>
          </table>
          </td>
          <td width="20"> </td>
        </tr>
      </tbody>
    </table>
      <?php
     include "bottom.php";
     ?>
```

文件 son_module_add.php 位于配套素材包的 ch21\manage\下，是子版块添加页面。具体代码如下。

```
    <?php
    include "session.inc";
    include "fun_head.php";
    head("子版块添加");
    include "../inc/mysql.inc";
    include "../inc/myfunction.inc";
    $aa=new mysqlconn;
    $bb=new myfunction;
```

```
    $aa->link("");
    $add_tag=@$_GET['add_tag'];
    if ($add_tag==1){
    $father_module_id=$_POST['father_module_id'];
    $module_name=$_POST['module_name'];
    $module_cont=$_POST['module_cont'];
    $user_name=$_POST['user_name'];
     if ($father_module_id=="" or $module_name=="" or $module_cont==""){
     echo "===对不起，您添加子版块不成功：<font color=red>隶属的父版块、
    子版块的名称和简介全不能为空</font>！===";
     }else{
     $query="insert into son_module_info(father_module_id,
    module_name,module_cont,user_name)
    values('$father_module_id','$module_name','$module_cont','$user_name')";
     $aa->excu($query);
     echo "===恭喜您，子版块添加成功！===";
     }
    }
    ?>
    <table width="100%" height="389" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
      <tbody>
        <tr>
          <td width="20"> </td>
          <td valign="top"><br />
            <form action="?add_tag=1" method="post" name="form1" id="form1">
              <table    width="408"    height="139"    border="0"    align="center"
cellpadding="0"
    cellspacing="1" bordercolor="#FFFFFF" bgcolor="449ae8">
                <tr bgcolor="#dcccccc">
                  <td width="94" height="25" bgcolor="e0eef5"><div align="right">
    隶属的父版块:</div></td>
                  <td              width="306"              bgcolor="e0eef5"><?php
$bb->father_module_list("");?></td>
                </tr>
                <tr bgcolor="#dddddd">
                  <td  height="25"  bgcolor="#FFFFFF"><div  align="right"> 子 版 块 名
称:</div></td>
                  <td      bgcolor="#FFFFFF"><input      type="text"      size="20"
name="module_name" /></td>
                </tr>
                <tr bgcolor="#dddddd">
                  <td height="25" align="right" valign="middle" bgcolor="#FFFFFF">
简介:</td>
                  <td bgcolor="#FFFFFF"><textarea name="module_cont"
```

```
    cols="42" rows="3"></textarea></td>
              </tr>
              <tr bgcolor="#dddddd">
                <td height="25" align="right" valign="middle" bgcolor="#FFFFFF">
版主用户名:</td>
                <td      bgcolor="#FFFFFF"><input      type="text"      size="20"
name="user_name" />
                  可不填。</td>
              </tr>
              <tr bgcolor="#dddddd">
                <td height="33" colspan="2" bgcolor="e0eef5"><div align="center">
                    <input name="submit" type="submit" value="提交" />
                        
                  <input name="reset" type="reset" value="重置" />
                </div></td>
              </tr>
            </table>
          </form>
          <br />
        <br /></td>
        <td width="20"> </td>
      </tr>
    </tbody>
  </table>
  <?php
   include "bottom.php";
  ?>
```

文件 sonr_module_bj.php 位于配套素材包的 ch21\manage\下，是子版块编辑页面。具体代码如下。

```
  <?php
  include "session.inc";
  include "fun_head.php";
  head("子版块管理>>编辑");
  include "../inc/mysql.inc";
  include "../inc/myfunction.inc";
  $aa=new mysqlconn;
  $bb=new myfunction;
  $aa->link("");
  $module_id=$_GET['module_id'];
  $update_tag=@$_GET['update_tag'];
  if ($update_tag==1){
  $father_module_id=$_POST['father_module_id'];
  $module_name=$_POST['module_name'];
  $module_cont=$_POST['module_cont'];
```

```
$user_name=$_POST['user_name'];
 if ($father_module_id=="" or $module_name=="" or $module_cont==""){
 echo "===对不起，您编辑子版块不成功：<font color=red>隶属的父版块、
子版块的名称和简介全不能为空</font>! ===";
 }else{
 $query="update son_module_info
set
father_module_id='$father_module_id',module_name='$module_name',module_cont
='$module_cont',user_name
='$user_name' where id='$module_id'";
 $aa->excu($query);
 echo "===恭喜您，编辑子版块添加成功！===";
 }
}
$query="select * from son_module_info where id='$module_id'";
$rst=$aa->excu($query);
$module=mysqli_fetch_array($rst,MYSQL_ASSOC);
?>
<table width="100%" height="389" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
   <tbody>
     <tr>
       <td width="20"> </td>
       <td valign="top"><br />
         <form action="?update_tag=1&module_id=<?php echo $module_id?>"
method="post" name="form1" id="form1">
           <table   width="408"   height="139"   border="0"   align="center"
cellpadding="0"
             cellspacing="1" bordercolor="#FFFFFF" bgcolor="449ae8">
             <tr bgcolor="#dcccccc">
               <td width="94" height="25" bgcolor="e0eef5"><div align="right">
隶属的父版块:</div></td>
               <td width="306"
bgcolor="e0eef5"><?php
$bb->father_module_list($module['father_module_id']);?></td>
             </tr>
             <tr bgcolor="#dddddd">
               <td  height="25"  bgcolor="#FFFFFF"><div  align="right"> 子 版 块 名
称:</div></td>
               <td      bgcolor="#FFFFFF"><input      type="text"      size="20"
name="module_name" value="<?php echo $module['module_name']?>" /></td>
             </tr>
             <tr bgcolor="#dddddd">
               <td height="25" align="right" valign="middle" bgcolor="#FFFFFF">
简介:</td>
```

```
                <td  bgcolor="#FFFFFF"><textarea  name="module_cont"  cols="42"
rows="3">
    <?php echo $module['module_cont']?></textarea></td>
              </tr>
              <tr bgcolor="#dddddd">
                <td height="25" align="right" valign="middle" bgcolor="#FFFFFF">
版主用户名:</td>
                <td     bgcolor="#FFFFFF"><input     type="text"     size="20"
name="user_name" value="
    <?php echo $module['user_name']?>" />
                  可不填。</td>
              </tr>
              <tr bgcolor="#dddddd">
                <td height="33" colspan="2" bgcolor="e0eef5"><div align="center">
                    <input name="submit" type="submit" value="提交" />
                        
                  <input name="reset" type="reset" value="重置" />
                </div></td>
              </tr>
            </table>
          </form>
          <br />
        <br /></td>
        <td width="20"> </td>
      </tr>
    </tbody>
    </table>
    <?php
     include "bottom.php";
    ?>
```

文件 son_module_list.php 位于配套素材包的 ch21\manage\下，是子版块显示页面。具体代码如下。

```
    <?php
    include "session.inc";
    include "fun_head.php";
    head("子版块管理");
    include "../inc/mysql.inc";
    include "../inc/myfunction.inc";
    $aa=new mysqlconn;
    $bb=new myfunction;
    $aa->link("");
    //////////删除子版块//////////////////////////////
    $del_tag=@$_GET['del_tag'];
    if ($del_tag==1){
     $module_id=$_GET['module_id'];
```

```
    $query="delete from son_module_info where id='$module_id'";
    $aa->excu($query);
    echo "==恭喜您，删除子版块信息成功！==<br />";
   }
   ///////////////////按显示顺序查询父版块信息表////////////
    $query="select * from father_module_info order by show_order";
    $rst=$aa->excu($query);
   ?>
   <table width="100%" height="390" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
     <tbody>
       <tr>
         <td width="20"> </td>
         <td valign="top"><br /><table width="80%" border="0" align="center"
   cellpadding="0" cellspacing="1" bgcolor="449ae8">
           <tr bgcolor="#cccccc">
             <td width="74" height="23" bgcolor="e0eef5"><div align="center">显示
序号</div></td>
             <td width="84" bgcolor="e0eef5"><div align="center">父版块名称
</div></td>
             <td width="410" bgcolor="e0eef5"><div align="center">子版块名称
</div></td>
             <td colspan="2" bgcolor="e0eef5"><div align="center">操作</div></td>
           </tr>
           <?php
        while($father_module=mysqli_fetch_array($rst,MYSQL_ASSOC)){
        ?>
           <tr>
             <td height="19" bgcolor="#FFFFFF"><div align="center">
   <?php echo $father_module['show_order']?></div></td>
             <td colspan="4" align="left" valign="middle" bgcolor="#CCCCCC">
   <?php echo $father_module['module_name']?></td>
             </tr>
         <?php
           ////////从子版块信息表中按 id 顺序查询隶属该父版块的子版块的信息/////////////
         $query="select * from son_module_info where
father_module_id='".$father_module['id']."'
   order by id";
         $rst2=$aa->excu($query);
         $m=0;
         while($son_module=mysqli_fetch_array($rst2,MYSQL_ASSOC)){
         $m++;
         ?>
           <tr>
             <td height="19" bgcolor="#FFFFFF"> </td>
```

```
            <td  align="center"  valign="middle"  bgcolor="#FFFFFF"><?php  echo
$m?></td>
            <td bgcolor="#FFFFFF"><?php echo $son_module['module_name']?></td>
            <td width="80" align="center"
   bgcolor="#FFFFFF"><a href="son_module_bj.php?module_id=
   <?php echo $son_module['id'];?>">编辑</a></td>
            <td       width="80"       align="center"       bgcolor="#FFFFFF"><a
href="?del_tag=1&module_id=
   <?php echo $son_module['id']?>">删除</a></td>
          <?php }?>
        </tr>
          <?php }?>
        </table>

        </td>
        <td width="20"> </td>
      </tr>
    </tbody>
   </table>
   <?php
       include "bottom.php";
   ?>
```

文件 user _list.php 位于配套素材包的 ch21\manage\下，是显示所有用户的页面。具体代码如下。

```
   <?php
   include "session.inc";
   include "fun_head.php";
   head("所有用户");
   include "../inc/mysql.inc";
   include "../inc/myfunction.inc";
   $aa=new mysqlconn;
   $bb=new myfunction;
   $aa->link("");
   //////////删除注册用户///////////////////////////////
   $del_tag=@$_GET[del_tag];
   if ($del_tag==1){
    $user_id=@$_GET[user_id];
    $query="delete from user_info where id='$user_id'";
    $aa->excu($query);
    echo "==恭喜您，删除注册用户信息成功！==<br />";
   }
   ////////////从用户信息表中查询所有用户/////////////////
    $query="select * from user_info order by id desc";
   ?>
   <table width="100%" height="390" border="0" cellpadding="0" cellspacing="0"
```

```
bgcolor="f0f0f0">
     <tbody>
       <tr>
         <td width="20"> </td>
         <td valign="top"><br />
           <table   width="80%"   border="0"   align="center"   cellpadding="0"
cellspacing="0">
             <tr>
               <td height="30" align="left"
   valign="middle"><?php $bb->page($query,@$page_id,@$add,20)?></td>
             </tr>
           </table>
           <table   width="80%"   border="0"   align="center"   cellpadding="0"
cellspacing="1" bgcolor="449ae8">
           <tr bgcolor="#cccccc">
             <td width="46" height="23" bgcolor="e0eef5"><div align="center">序号
</div></td>
             <td   width="213"   bgcolor="e0eef5"><div   align="center">  用  户  名
</div></td>
             <td  width="201"  bgcolor="e0eef5"><div  align="center"> 注 册 时 间
</div></td>
             <td  width="188"  bgcolor="e0eef5"><div  align="center">最后登录时间
</div></td>
             <td width="80" bgcolor="e0eef5"><div align="center">操作</div></td>
           </tr>
   <?php
    $rst=$aa->excu($query);
    $m=0;
    while($user=mysqli_fetch_array($rst,MYSQL_ASSOC)){
    $m++;
   ?>
           <tr>
             <td  height="19"  bgcolor="#FFFFFF"><div  align="center"><?php  echo
@$m;?></div></td>
             <td      bgcolor="#FFFFFF"><div      align="center"><?php      echo
$user['user_name']?></div></td>
             <td      bgcolor="#FFFFFF"><div      align="center"><?php      echo
$user['time1']?></div></td>
             <td align="center" bgcolor="#FFFFFF"><?php echo $user['time2']?></td>
             <td align="center" bgcolor="#FFFFFF"><a href="?del_tag=1&user_id=
   <?php echo $user['id']?>">删除</a></td>
             </tr>
           <?php }?>
         </table>
           <table   width="80%"   border="0"   align="center"   cellpadding="0"
```

```
cellspacing="0">
            <tr>
              <td height="30" align="right" valign="middle">
            <?php
            $query="select * from user_info order by id desc";
            $bb->page($query,@$page_id,@$add,20);
            ?></td>
            </tr>
          </table></td>
        <td width="20"> </td>
      </tr>
    </tbody>
  </table>
  <?php
   include "bottom.php";
  ?>
```

文件 user_js.php 位于配套素材包的 ch21\manage\下，是用户的检索页面。具体代码如下。

```
  <?php
  include "session.inc";
  include "fun_head.php";
  head("用户检索");
  include "../inc/mysql.inc";
  include "../inc/myfunction.inc";
  $aa=new mysqlconn;
  $bb=new myfunction;
  $aa->link("");
  //////////删除用户///////////////////////////////
  $del_tag=@$_GET[del_tag];
  if ($del_tag==1){
   $del_id=@$_GET[del_id];
   $query="delete from user_info where id='".$del_id."'";
   $aa->excu($query);
   echo "==恭喜您，删除用户信息成功，请继续！==<br />";
  }
  //////////////////////////////////////////////////
   $user_name=@$_POST[user_name];
  ?>
  <table width="100%" height="390" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
    <tbody>
      <tr>
        <td width="20"> </td>
        <td valign="top"><br />
          <table    width="70%"    border="0"    align="center"    cellpadding="0"
```

```
cellspacing="1" bgcolor="449ae8">
          <tr bgcolor="#cccccc">
           <form id="form1" name="form1" method="post" action="?">
         <td height="23" bgcolor="e0eef5"><div align="center">
               <input type="text" name="user_name" size="16" value="
   <?php echo $user_name?>" />    
               <input type="submit" name="Submit" value="提交" />
             </div></td>
           </form>
         </tr>
           <tr>
           <td height="19" align="center" bgcolor="#FFFFFF">
         <?php
         if ($user_name==""){
           echo "请输入您要检索的用户名，点提交查询！";
         }else{
           $query="select * from user_info where user_name='$user_name'";
           $rst=$aa->excu($query);
           if (mysqli_num_rows ($rst)==0){
               echo "很抱歉，系统没有检索到您要查找的用户！";
           }else{
               $user=mysqli_fetch_array($rst,MYSQL_ASSOC);
         ?>
         <table width="100%" border="0" cellspacing="2" cellpadding="0">
             <tr>
               <td width="29%" height="20" align="right" bgcolor="#dddddd">用户
名:</td>
               <td   width="71%"   align="left"   bgcolor="#dddddd"><?php   echo
$user['user_name']?></td>
             </tr>
             <tr>
               <td height="20" align="right" bgcolor="#dddddd">登录口令:</td>
               <td        align="left"        bgcolor="#dddddd"><?php        echo
$user['user_pw']?></td>
             </tr>
             <tr>
               <td height="20" align="right" bgcolor="#dddddd">注册时间:</td>
               <td        align="left"        bgcolor="#dddddd"><?php        echo
$user['time1']?></td>
             </tr>
             <tr>
               <td height="20" align="right" bgcolor="#dddddd">最后登录时间:</td>
               <td        align="left"        bgcolor="#dddddd"><?php        echo
$user['time2']?></td>
             </tr>
```

```
            <tr>
              <td height="20" colspan="2" align="center"
bgcolor="#dddddd"><a href=?del_tag=1&del_id=<?php echo $user['id']?>>
删除此用户</a></td>
              </tr>
           </table>
         <?php
           }
         }
         ?>
         </td>
           </tr>
        </table>
        </td>
        <td width="20"> </td>
      </tr>
    </tbody>
</table>
  <?php
 include "bottom.php";
  ?>
```

文件 user_pw_change.php 位于配套素材包的 ch21\manage\下，是管理员密码修改页面。具体代码如下。

```
<?php
include "session.inc";
include "fun_head.php";
head("密码更改");
include "../inc/mysql.inc";
include "../inc/myfunction.inc";
$aa=new mysqlconn;
$bb=new myfunction;
$aa->link("");
//////////更改密码//////////////////////////////
$tijiao=$_POST[tijiao];
if ($tijiao=="提 交"){
 $pw_old=$_POST[pw_old];
 $pw_new1=$_POST[pw_new1];
 $pw_new2=$_POST[pw_new2];
 if ($pw_new1!=$pw_new2){
     echo "===您两次输入的新密码不一致,请重新输入!===";
 }else{
     $query="select * from manage_user_info where user_name='
$_SESSION[manage_name]' and user_pw='$pw_old'";
```

```
                $rst=$aa->excu($query);
        if (mysqli_num_rows ($rst)==0){
            echo "===您输入的旧密码不正确,请重新输入!===";
        }else{
            $query="update manage_user_info set
    user_pw='$pw_new1' where user_name='$_SESSION[manage_name]'";
            $aa->excu($query);
            echo "===恭喜您,您的登录密码修改成功!===";
        }
    }
    }
    ?>
    <table width="100%" height="390" border="0" cellpadding="0" cellspacing="0"
bgcolor="f0f0f0">
      <tbody>
        <tr>
          <td width="20"> </td>
          <td valign="top"><br />
            <form id="form1" name="form1" method="post" action="#">
            <table    width="60%"    border="0"    align="center"    cellpadding="0"
cellspacing="1" bgcolor="449ae8">
            <tr bgcolor="#cccccc">
            <td width="27%" height="23" align="right" bgcolor="e0eef5">用户名:</td>
            <td width="73%" align="left" bgcolor="e0eef5">
    <?php echo $_SESSION[manage_name]?></td>
            </tr>
              <tr>
              <td height="19" align="right" bgcolor="#FFFFFF">原口令:</td>
              <td align="left" bgcolor="#FFFFFF"><input type="text" name="pw_old"
size="16" /></td>
              </tr>
              <tr>
                <td height="19" align="right" bgcolor="#FFFFFF">新密码:</td>
                <td       align="left"      bgcolor="#FFFFFF"><input      type="text"
name="pw_new1" size="16" /></td>
              </tr>
              <tr>
                <td height="19" align="right" bgcolor="#FFFFFF">再次新密码:</td>
                <td       align="left"      bgcolor="#FFFFFF"><input      type="text"
name="pw_new2" size="16" /></td>
              </tr>
              <tr>
                <td height="19" colspan="2" align="center" bgcolor="#FFFFFF">
    <input type="submit" name="tijiao" value="提 交" />
                   
```

```
            <input type="reset" name="Submit2" value="重 置" /></td>
            </tr>
          </table>
      </form>
          </td>
          <td width="20"> </td>
        </tr>
      </tbody>
    </table>
      <?php
    include "bottom.php";
      ?>
```

成功登录到网站后台管理系统的效果如图 21-9 所示。

单击页面左侧的【论坛版块管理】选项，即可在展开的列表中查看版块管理操作，如图 21-10 所示。

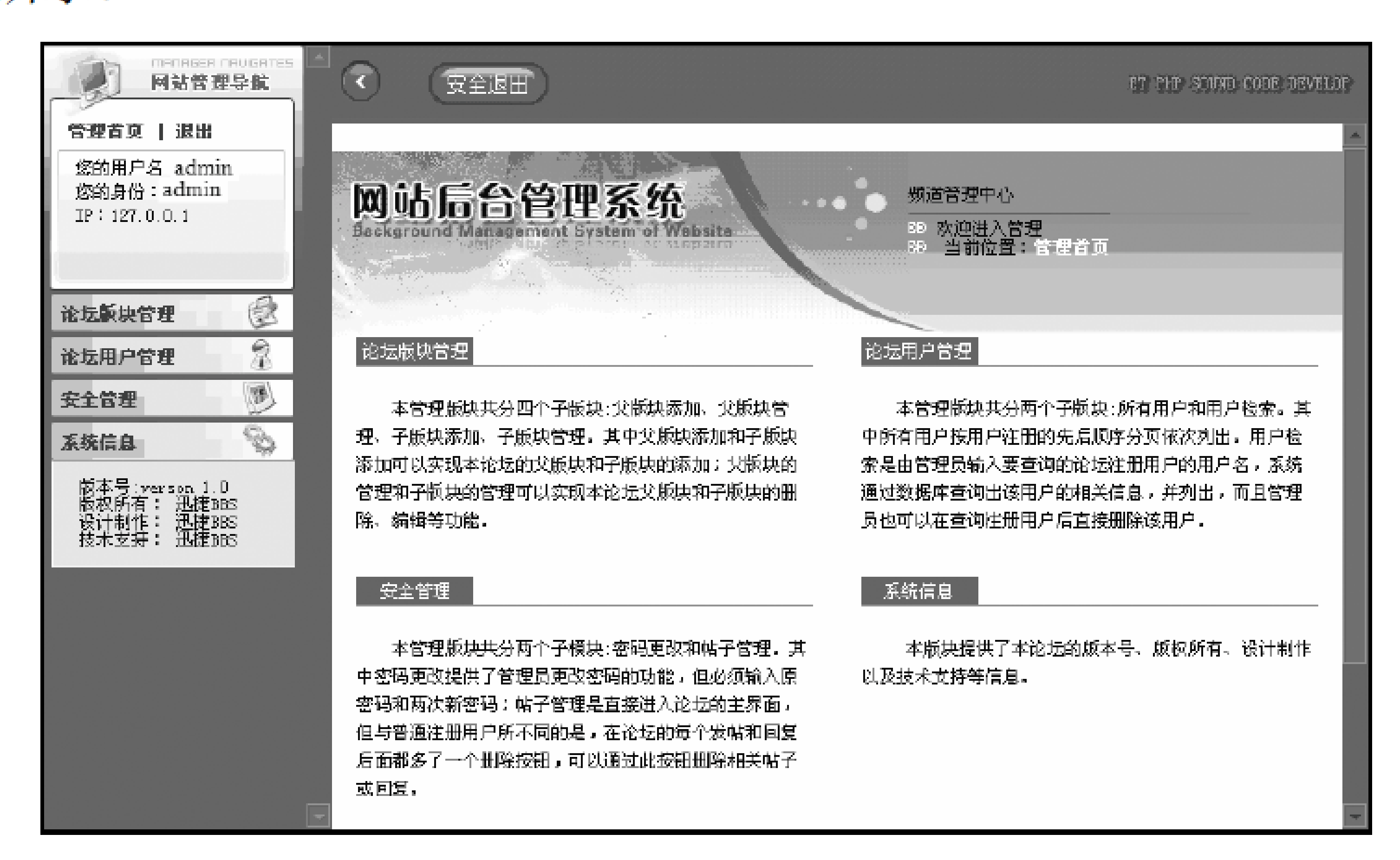


图 21-9　后台管理系统主页面

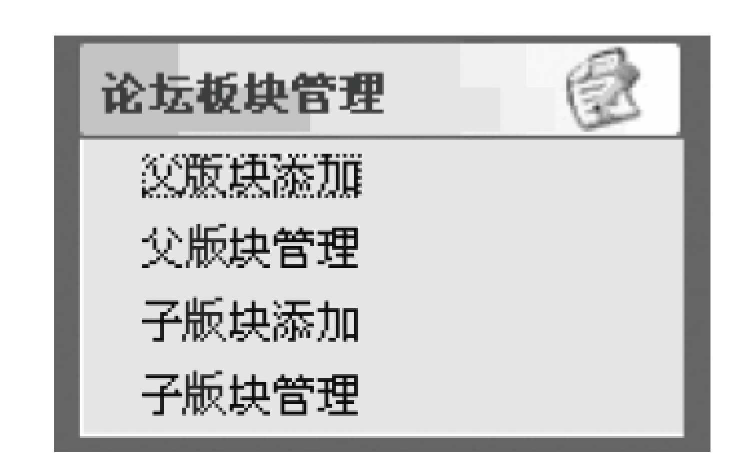


图 21-10　【论坛版块管理】列表

单击【父版块添加】选项，即可在右侧的窗口中输入显示序号和父版块名称，然后单击【提交】按钮，如图 21-11 所示。

添加成功后，即可显示成功提示信息，如图 21-12 所示。



图 21-11　添加父版块



图 21-12　添加父版块成功提示信息

刷新论坛的主页面，即可看到新添加的父版块，如图 21-13 所示。

在后台管理主页面的【论坛版块管理】列表中，单击【父版块管理】选项，即可进入父版块管理页面中，如图 21-14 所示。用户可以编辑和删除存在的父版块。



图 21-13　查看父版块



图 21-14　编辑父版块

在后台管理主页面的【论坛版块管理】列表中，单击【子版块添加】选项，即可进入子版块添加页面，如图 21-15 所示。用户输入相关信息后，单击【提交】按钮即可。

刷新论坛的主页面，即可看到新添加的子版块，如图 21-16 所示。



图 21-15　添加子版块　　　　图 21-16　查看添加的子版块

单击页面左侧的【论坛用户管理】选项，即可在展开的列表中查看用户管理操作，如图 21-17 所示。

单击【所有用户】选项，即可在右侧的窗口中查看论坛的用户，如图 21-18 所示。

论坛用户管理
所有用户
用户检索

图 21-17 【论坛用户管理】列表



图 21-18 【所有用户】页面

在后台管理主页面的【用户版块管理】列表中，单击【用户检索】选项，进入用户检索页面，输入用户名后，单击【提交】按钮，即可显示用户的具体信息，如图 21-19 所示。

在后台管理主页面中，单击页面左侧的【安全管理】选项，即可在展开的列表中管理密码和帖子操作，如图 21-20 所示。



图 21-19 用户检索页面

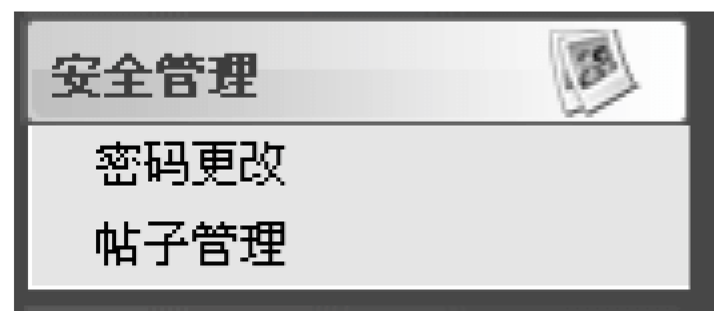


图 21-20 【安全管理】列表

单击【密码更改】选项，即可在右侧的页面中修改用户的密码，如图 21-21 所示。

用户名: aaa
原口令:
新密码:
再次新密码:
提 交 重 置

图 21-21 更改用户密码页面